硕士学位论文

基于虚拟物理仿真思考的开放任务求解

PHYSICAL SIMULATION AND REASONING BASED TASK-AGNOSTIC LEARNING

柴士童

哈尔滨工业大学

2020 年 12 月

国内图书分类号：TM301.2　　学校代码：10213
国际图书分类号：62-5　　密级：公开

工程硕士学位论文

基于虚拟物理仿真思考的开放任务求解

硕士研究生：柴士童

导　　师：范晓鹏

申请学位：工程硕士

学　　科：软件工程

所在单位：计算学部

答辩日期：2020 年 12 月

授予学位单位：哈尔滨工业大学

Classified Index: TM301.2
U.D.C: 62-5

Dissertation for the Master's Degree in Engineering

PHYSICAL SIMULATION AND REASONING BASED TASK-AGNOSTIC LEARNING

Candidate: CHAI Shitong
Supervisor: Professor FAN Xiaopeng
Academic Degree Applied for: Master of Engineering
Specialty: Software Engineering
Affiliation: Faculty of Computing
Date of Defence: December, 2020
Degree-Conferring-Institution: Harbin Institute of Technology

摘　要

本文在Pyrobolearn机器人仿真环境中设计了一个要求机械臂的末端执行器到达指定物体附近的开放任务，并设计了新的强化学习算法用于训练智能体在未获得任务相关的奖励之前在未知环境中学习到可泛化到该开放任务的策略。

在 TD3 算法和 HER 算法的基础上，本文引入了基于局部敏感哈希和计数的奖励用于鼓励智能体探索未知环境，并提出了轨迹替换方法。

本文提出的斯混合噪声层被用于提供自适应的策略噪声。提出的基于物理仿真引擎仿真时间的奖励被用于在未知任务的环境中鼓励智能体学习有效可泛化的策略，并在获得稀疏奖励的开放任务后快速适应到新的策略。

关键词：元学习；强化学习；机器人学；物理仿真；深度学习

Abstract

A task which requires the end effector of a manipulator to reach a specific body is designed with the help of the robot learning framework Pyrobolearn. New reinforcement learning algorithms are proposed to train an agent to learn in a task-agnostic environment without reward related to the task, where the learnt policy of the agent should be generalizable to the proposed task.

Based on TD3 and HER algorithms, a locality sensitive hashing based counting reward is introduced to encourage the agent to explore in the unknown environment. And a trajector replacement strategy is proposed.

A mixed gaussian noise layer is proposed to provide a adaptive policy noise. A reward based on physics simulation time is proposed to encourage the agent to learn a generalizable policy in a task agnostic environment, and adapt to a new policy after given a task setting with sparse reward.

Keywords: Meta Learning, Reinforcement Learning, Robotics, Physics Simulation, Deep Learning

目　录

第 1 章 绪论

1.1 选题背景及研究意义

1.1.1 选题背景

一直以来，强化学习和机器人学都是人工智能研究中的热门领域。在 2008 年，Deepmind 团队基于深度强化学习研发的围棋人工智能系统 AlphaGo Zero 在零知识自我对弈的情况下在几天之内超越了旧的系统 AlphaGo，而 AlphaGo 曾击败了围棋领域中世界公认的专家柯洁等人[1]。Deepmind 的研究激发了各个研究领域对于强化学习技术的兴趣。事实上，强化学习已经成为最热门的研究领域之一，并在自动控制、运筹学、机器人学、游戏智能体和无人驾驶等领域中获得了广泛的应用[2]。在这些领域中，机器人学是和前沿强化学习算法关系最密切的领域之一。传统的机器人如机械臂、四足机器人等，可以用强化学习训练得到的智能体进行控制，并在与环境交互过程中根据环境反馈使策略得到进一步的优化。

然而，由于机器人设计和制造的成本较高，通用的多关节机器人通常非常昂贵。而且机器人通常容易在强化学习中的各种随机探索中受到损坏，并导致控制系统和智能体策略在训练过程中出现错误。因此在实际的机器人上进行所有强化学习训练是不现实的[3,4]。为了避免这个问题，可以使用物理仿真引擎对机器人和环境进行建模，并在实际测试智能体之前先在仿真环境中对智能体进行训练。幸运的是，随着强化学习仿真需求的增加，越来越多的针对机器人的仿真环境开始出现，在这些仿真环境中，可以像在真实环境中一样控制机器人的关节、调节各种参数，或获得传感器数据等等，并可以做到在真实环境中难以做到的设定复杂的稀疏奖励、获取碰撞次数、和改变环境的物理参数等操作[5]。

虽然已经有大量的软件系统可以用于在一个环境建模完全精确的情形下解决一个良定义的任务[3]，如何让智能体在面对未知的新环境和未知的新任务后能够有效泛化之前学习到的策略仍然是一个未完全解决的问题[6,7]。人类可以在陌生的环境中用很少次数的探索自然地掌握大量有效信息，还可以利用已有经验对大量物体进行分类、提高对物理实体运动的预测能力，或创新性地设计工具解决问题。由于人类大脑的思维活动难以进行定量研究，这个过程通常

很难被数值化为一个单一的奖励函数或强化学习算法。

本课题致力于解决上述问题，即设计算法从而可以训练出能在任务奖励未知的环境中进行探索，获取环境信息，学习基本策略，并在任务确定后快速调节旧策略以适应新任务的智能体。为了实现这个目标，需要利用现有的开源物理仿真引擎、前沿的强化学习算法和具有强大函数拟合能力的深度神经网络来设计新算法。

1.1.2 研究意义

机器人控制对于工业制造有着重要的意义，在工厂流水线上，机械臂常常被设计为只能完成单个简单任务，或需要工人远程控制。虽然它们已经极大地提高了生产效率，减少了对工人们生命财产安全的威胁，但是机械臂由于价格昂贵，仅仅用于单一任务会造成极大的资源浪费。

本课题提出的算法有希望训练出可以在对未知任务奖励的环境进行充分探索之后，快速适应多种不同任务的机器人智能体，从而扩大现有强化学习算法的适用范围，解决更复杂的控制任务，增强机器人智能体泛化策略的能力。

此外，本课题还可以加深对现有强化学习算法在机器人控制中应用价值的理解，可以通过机器人仿真和控制帮助提前发现在应用算法到实际机器人控制时可能出现的问题，可以通过设计和调整深度神经网络进一步了解不同结构的神经网络对智能体性能的影响。

1.2 国内外研究进展

国内外已经有了很多关于让机器人智能体使用工具和泛化已学习的策略到新任务的研究。其中基于引导视觉的工具使用、无监督兴趣导向的探索、模仿学习和自驱动的主动学习与本课题有关。

1.2.1 基于引导视觉预见的工具使用

引导视觉预见[8] 可以使机械臂智能体从人类演示中学习并泛化学习到的能力以在不同的环境中使用工具。这种方法包含动作提案模型和预测模型。其中动作提案模型使用演示动作数据来训练一个自回归的长短时记忆网络模型来根据图像传感器拍摄到的图像数据生成动作。预测模型是基于卷积长短时记忆网络的[9]。预测模型被用于对物理实体的运动进行物理预测，并可以被用于筛选不能完成指定目标的动作序列。训练过程分为基于演示的模仿学习和利用握爪反射的随机自动训练。在测试过程中，指定的目标被定义为像素移动，预

测模型输出的像素位置和真实像素位置的距离被用于评估动作好坏。在指定任务后，动作序列从动作提案模型中采样，并使用交叉熵方法结合预测模型进行优化。实验表明，此种方法可以比单纯模仿学习在新环境中获得更好的泛化性能。

1.2.2 无监督兴趣导向探索

引导视觉预见需要有人类使用工具的演示数据才能正常工作，而对于更一般的环境，人类的演示数据可能是无法获得的，此时智能体应当可以在没有指定任务的情况下在环境中探索。为了解决这个问题，一个兴趣导向的探索方法被提出了[10]。这个方法结合了解纠缠目标空间的变分自编码机（VAE)[11] 和兴趣导向的 IMGEP（Intrinsically Motivated Goal Exploration Process）方法[12]。在目标空间被给定后，IMGEP 框架下的智能体会倾向于选择更有可能增加竞争力的目标。因为不像通常的强化学习算法一样在给定单一目标后做训练，而是无监督地选择目标进行训练，因此这是一个元策略算法。在探索过程中，智能体会学习到被 β-VAE 解纠缠后的观测，因此智能体可以分离地探索不同的物体。在实验中，此种方法获得了比单纯 IMGEP 方法更大的探索率。

1.2.3 模仿学习

在需要使用工具求解的开放任务中，往往有着稀疏奖励，随机探索很难刚好完成一个完整的动作序列，最终成功地使用工具完成任务，并获得奖励。模仿学习通过人类的专家策略，可以极大地加速这种随机探索过程。通过人类提供的演示数据，智能体应当能够学习到更好的探索策略，增大获得奖励的概率。相关研究表明，使用常规的强化学习算法和运动剪辑数据集，智能体可以学会组合学到的不同的技能，并用于求解多种任务[13]。不仅如此，智能体也可以从视频中学习到一些技能[14]。这意味着现有模仿学习方法可以利用网络中大量的视频数据进行学习。

1.2.4 自驱动的主动学习

在开放任务求解中，主动学习技术也可以被使用[15? –17]。为了在主动学习过程中获得预测物理环境的能力，可以使用带策略的循环 Q 网络来减少未知物理性质的熵[18]。在机器人学中，逆动力学模型对于稳健控制非常重要。一种主动学习方法使用竞争力来选择目标并在高维连续空间中学习各种技能，并用于机器人控制[19]。仿真结果表明这种方法能够帮助机器人智能体探索到随机策略

难以探索到的区域。

1.3 研究内容和方法

1.3.1 研究内容

在开放任务中，时刻 t 下会有一个稀疏奖励 $R_t \in \{0,-1\}$ 被提供给智能体。奖励为-1 表示任务失败，奖励为 0 表示任务成功。该奖励是关于环境状态的函数，即 $reward : \mathcal{S} \rightarrow \{0,-1\}$，其中 $\mathcal{S}$ 表示状态空间。智能体被允许在 $t < T_{start}$ 时间内对环境进行探索，但是无法获知函数 $reward$ 或由此函数计算得到的稀疏奖励 R_t，只能通过它观测到的状态向量 $s \in \mathcal{S}$ 来计算内部奖励（intrinsic reward），根据此奖励对环境进行探索，并学习能用于被 $reward$ 函数表示的任务有关的技能。在 $t \geq T_{start}$ 时，在每一时刻 t 对智能体提供由函数 $reward$ 计算得到的稀疏奖励 $R_t = reward(s)$。理想情况下，在 $t < T_{start}$ 时使用特定策略 π 进行探索的智能体应当能比使用随机选择动作的策略 π 进行探索的智能体更快地收敛到更高的奖励，即 $\mathbb{E}_{\pi}[R_t] > \mathbb{E}_{\pi'}[R_t]$。

1.3.2 研究方法

本文的实验环境是基于 Gym[20] 和 Pyrobolearn[21] 搭建的，其中 Gym 用于初步验证算法的正确性，大部分仿真实验在 Pyrobolearn 中完成。实验中对神经网络的优化使用了 Pytorch[22]，算法主体程序使用了 Python 语言。

本文的研究方法是基于内部奖励的强化学习。其中内部奖励被用于鼓励智能体在未知环境中探索，并掌握可用于解决开放任务的技能，它与任务特定的稀疏奖励 $reward$ 无关。

在本文的研究中，内部奖励与事后经验重放（Hindsight Experience Replay）被结合起来，用于获得更丰富的探索奖励。其中，多种内部奖励方法被使用，如基于局部敏感哈希的计数奖励、基于正向动力学预测的奖励和本文提出的物理仿真时间奖励。本文中的强化学习架构采用传统的演员 - 评论家（Actor-Critic）模式[23]，其中演员网络和评论家网络都使用了多层感知机。为了加速算法的收敛，减小训练过程中的不稳定性，训练算法借用了 TD3 算法[24] 的结构，引入了两个评论家，设计了靶网络以减缓演员网络和评论家网络参数更新的速度，并提出了混合高斯噪声层用于防止确定性策略收敛到局部极小。

第 2 章 相关算法基础

本文中提出的算法使用了大量最新强化学习算法中的思想，因此有必要介绍强化学习中常用的概念和现有强化学习算法的思想。

2.1 强化学习基础

在强化学习中，我们对智能体如何与环境交互感兴趣，这涉及到 4 个概念：模型、状态、动作和奖励[25]。由与智能体和环境相关的可观测量构成了智能体的状态向量 s，它表示了智能体当前在环境中所处的状态。每个状态从状态空间 $\mathcal{S}$ 中取值，$\mathcal{S}$ 包含了所有智能体和环境可以取到的状态。一个智能体可以在某一时刻做出动作 a 并因此转移到下一个状态。所有的动作从动作空间 $\mathcal{A}$ 中取值。如果完全已知一个环境的模型，且已知智能体的当前状态，就可以知道智能体做出任一动作的效果。当智能体做出某一个动作之后，根据当前的任务和环境可以得到一个奖励 r，它表示智能体从环境中得到的反馈，并从奖励空间 $\mathcal{R}$ 中取值。状态转换、动作和它导致的奖励构成了一个轨迹。如果用 S_t，A_t，R_t 分别表示在时刻 t 下智能体的状态、动作和奖励，那么轨迹就可以表示为一个序列 $S_t, A_t, R_{t+1}, S_{t+1}, \cdots, S_T$，其中 T 是关心的整个片断结束的时刻。策略 $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$ 是智能体在当前状态 $S_t = s$ 时采取动作 $A_t = a$ 的条件概率。每个状态都有一个关联的值表示当前状态在特定任务下的价值，用状态价值函数 $V_\pi : \mathcal{S} \to \mathbb{R}$ 来表示从一个状态到这个值的映射。状态价值函数实际上表示了智能体使用当前策略 π 来选择动作，在可能获得奖励的多少。类似地，用动作价值函数 $Q_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ 表示智能体在给定当前状态和动作后，随后一直使用策略 π 来选择动作，在未来获得奖励的多少。在在实际算法中，为了防止对价值函数的拟合发散，需要给定一个折扣系数 $\gamma \in [0, 1]$ 来减少较远未来获得的奖励对当前价值函数的影响。强化学习的最终目标实际上为了最大化累积奖励：

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

有了累积奖励的定义后，状态价值函数 V_π 就可以写成在未来使用策略 π 的期望的累积奖励：

$$V_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$$

类似地，动作价值函数可以写成：

$$Q_\pi(s,a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$$

2.2 TD3 算法

在通常的演员 - 评论家模式的强化学习算法中[23]，演员网络被用于拟合最优的确定性策略 $\mu^*: \mathcal{S} \to \mathcal{A}$，评论家网络被用于拟合最优的动作价值函数 $Q^*: \mathcal{S} \times \mathcal{A} \to \mathbb{R}$。在给定一个迁移 (S_t, A_t, R_t, S_{t+1})，和现有的演员网络 μ 和评论家网络 Q 之后，根据如下损失函数对网络参数进行优化：

$$L_Q = \mathbb{E}[(R_t + \gamma Q(S_{t+1}, \mu(S_{t+1})) - Q(S_t, A_t))^2]$$

$$L_\mu = \mathbb{E}[-Q(S_t, \mu(S_t))]$$

但是由于在上述函数拟合过程中，评论家网络往往倾向于输出更高的动作值或状态价值。根据 TD3 算法[24]，可以使用两个评论家网络 $Q_1, Q_2: \mathcal{S} \times \mathcal{A} \to \mathbb{R}$。这两个评论家网络同时对未来时刻 $t+1$ 时刻的动作和状态进行评估并分别输出价值 $Q_1(S_{t+1}, A_{t+1})$ 和 $Q_2(S_{t+1}, A_{t+1})$。取这两个值的最小值作为对未来时刻 $t+1$ 的价值预测，即可得到修正后的评论家网络的损失函数：

$$L_{Q_1} = \mathbb{E}[(R_t + \gamma \min\{Q_1(S_{t+1}, \mu(S_{t+1})), Q_2(S_{t+1}, \mu(S_{t+1}))\} - Q_1(S_t, A_t))^2]$$

$$L_{Q_2} = \mathbb{E}[(R_t + \gamma \min\{Q_1(S_{t+1}, \mu(S_{t+1})), Q_2(S_{t+1}, \mu(S_{t+1}))\} - Q_2(S_t, A_t))^2]$$

在对演员网络进行优化时，只使用评论家网络 Q_1：

$$L_\mu = \mathbb{E}[-Q_1(S_t, \mu(S_t))]$$

为了防止在训练评论家网络时出现发散或损失不稳定的情况，可以对每个网络引入一个靶网络。对每个靶网络权重，不使用损失函数对其进行优化，而是使用原网络和靶网络权重的指数滑动平均进行更新。对上述网络 μ, Q_1, Q_2，有对应的靶网络 μ', Q_1', Q_2'。在训练刚开始时，保持原网络和靶网络权重相同。在之后的训练过程中，使用上述损失函数对 μ, Q_1, Q_2 进行更新，而使用如下公式分别对靶网络的权重 $W_{\mu'}, W_{Q_1'}, W_{Q_2'}$ 进行更新：

$$W_{\mu'} = \tau W_\mu + (1-\tau) W_{\mu'}$$

$$W_{Q_1'} = \tau W_{Q_1} + (1-\tau) W_{Q_1'}$$

$$W_{Q_2'} = \tau W_{Q_2} + (1-\tau) W_{Q_2'}$$

其中 τ 叫做 polyak 系数，它控制着每次权重更新的多少。TD3 算法工作原理如图2-1所示，图中的替换使用了上述指数滑动平均方法，L_μ 表示演员网络的损失，L_{Q_1} 和 L_{Q_2} 分别表示根据上述公式计算出的评论家网络 1 和评论家网络 2 的损失。

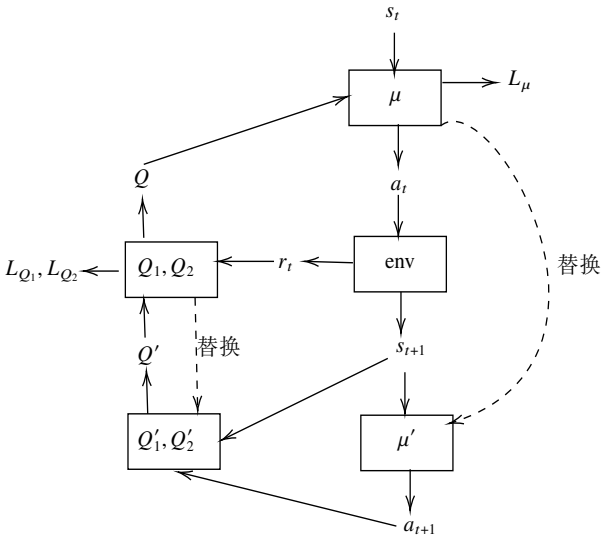

图 2-1 TD3 算法工作原理示意图

2.3 事后经验重放算法

事后经验重放（HER）算法[26]是一个用于稀疏奖励的强化学习算法，它主要是为了解决奖励过少导致的训练速度过慢的问题。

事后经验重放要求状态向量 s 由观测向量 o，已完成目标 g^a，期望目标 g^d 构成，即 $s = (o, g^a, g^d)$。在智能体与环境交互的过程中，每当获得一个迁移 (s_t, a_t, r_t, s_{t+1}) 后，就把它放入重放经验缓冲（replay buffer）中。通常情况下，在训练时，从重放经验缓冲中直接取出一个个迁移并根据这些数据对网络进行训练。

而在事后经验重放算法中，为了增加成功奖励的个数，对每个取出的属于片段 e_i 的迁移 j，也即 (s_j, a_j, r_j, s_{j+1})，以一定概率 p_{future} 取出在它之后的一个迁移 k，也即 (s_k, a_k, r_k, s_{k+1})，其中有 $j \le k \le T$，T 是片段 e_i 的长度，并把迁移 j 的状态向量 $s_j = (o_j, g_j^a, g_j^d), s_{j+1} = (o_{j+1}, g_{j+1}^a, g_{j+1}^d)$ 中的期望目标 g_j^d, g_{j+1}^d 替换为未来在 k 时刻已完成的目标 g_k^a，最后根据新的状态对这些替换过的迁移重

新计算奖励。

在上述算法中，概率 $p_{future} = 1 - \frac{1}{1+K_{replay}}$，其中 K_{replay} 是一个超参数，它决定了在经验重放时替换过的迁移所占比例的大小。

2.4 基于局部敏感哈希和计数的探索奖励

在开放任务中，由于奖励是稀疏的，智能体在自由探索过程中很难通过成功的奖励获知有关完成任务目标的信息，为了解决这一问题，需要引入任务无关的内部奖励来帮助智能体更快地探索有意义的状态，防止智能体对访问过的状态进行多次重复访问。

基于局部敏感哈希和计数的探索奖励[27–29] 使用一个局部敏感哈希函数对状态空间进行离散化，并对访问过的状态对应的哈希值进行计数，并通过此计数值计算出的奖励鼓励智能体访问之前未访问过的状态。

给定一个状态向量 $s \in \mathcal{S}$ 和一个动作向量 $a \in \mathcal{A}$，进行链接可以得到新向量 $x = (s, a)$，接着使用 SimHash 函数计算它的哈希值：

$$\phi(s) = \mathrm{sgn}(Ax) \in \{-1, 1\}^k$$

其中 A 是随机生成的矩阵，满足：

$$A \in \mathbb{R}^{k \times \dim(\mathcal{S})}, A_{ij} \sim \mathcal{N}(0, 1)$$

A 在算法的初始化过程中随机生成，并在之后的训练过程中保持不变，而 k 是一个超参数，它的大小决定了哈希向量的长度，因此决定了离散化后的状态空间的粒度。

每当智能体到达一个状态 s 后，都可以根据上述哈希函数计算哈希值 $\phi(s)$，并使用一个字典对此哈希值进行计数，即：

$$freq[\phi(s)] += 1$$

因此字典 $freq$ 就记录了访问具有同样局部敏感哈希值的相似状态的频率，并可计算奖励：

$$reward_{lsh}(s) = \frac{1}{\sqrt{freq[\phi(s)]}}$$

2.5 基于正向动力学预测的探索奖励

每当智能体获得一个迁移 (s_t, a_t, r_t, s_{t+1})，可用于进行正向动力学预测的信

息就更多了，这种信息蕴含在 $(s_t, a_t) \mapsto s_{t+1}$ 的映射中，它反映了环境的动力学性质。

基于正向动力学预测的探索奖励[30] 使用一个神经网络来拟合这种正向的动力学过程，即在理想情况下，预测模型 $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ 应当可以正确地预测未来的状态：$f(s_t, a_t) = s_{t+1}$。

对于经常访问到的简单状态和动作，正向动力学预测模型更有可能输出正确的预测值，而对于复杂未知的状态和动作，正向动力学预测模型则更有可能出错。这意味着可以把正向动力学预测模型的损失作为探索奖励来鼓励智能体探索更难被模型拟合的动力学过程。

给定一个迁移 (s_t, a_t, r_t, s_{t+1}) 和一个正向动力学预测模型 f，它的损失定义为：

$$L_{phy} = ||f(s_t, a_t) - s_{t+1}||^2$$

根据以上分析，可以把这个损失值当作内部奖励提供给智能体，于是有奖励函数：

$$reward_{phy}(s_t, a_t, s_{t+1}) = L_{phy}(s_t, a_t, s_{t+1})$$

第 3 章 实验原理与系统设计

3.1 Gym 和 Mujoco

Gym[20] 是一个由 OpenAI 团队开源的强化学习仿真环境，它已经成为了评价最先进的基线强化学习算法或机器人控制算法的标准环境。Mujoco[4] 是一个著名的商业物理仿真引擎，它在强化学习和机器人学中得到广泛应用，并成为 Gym 中机器人相关的环境的默认仿真引擎。在接下来的几章中，Gym 将被用来验证算法正确性，或被用于更方便地与其他现有算法进行对比。

Gym 提供一个 Python 语言的接口并且已经被包含在了 Python 包检索中。Gym 需要使用 OpenGL 的开源库 GLFW 来进行渲染。它可以直接使用 Python 包管理器 *pip* 进行安装。在 Gym 中，有很多内置的环境，包含大量著名的和强化学习有关基本任务可用于验证强化学习算法，并提供了统一的接口。例如，它包含一个经典的名为“推车杆”的控制任务，这个任务要求智能体通过水平地移动推车来平衡一个底部用自由活动的关节连接到推车上的杆。一个正常的强化学习算法应当可以在较长时间内避免杆失去平衡并倒下。

Mujoco（Multi-Joint dynamics with Cotact）是一个致力于仿真复杂关节运动、碰撞和多物体接触的物理引擎。它集成了粒子系统仿真、约束求解器、有限元积分器和凸优化器等等工具。它使用 ANSI C 编写并有一个 Python 的 API 封装。在 Gym 中，*FetchReach-v1* 环境需要使用 Mujoco 作为仿真器，因此在实验中 Mujoco、OpenGL 和 Gym 等工具都被安装在 Ubuntu 20.04 系统中，并设置了环境变量以保证正确的动态链接库 *libGL.so* 和 Mujoco 的二进制分发被加载。

一个 *FetchReach-v1* 环境的截图展示在图3-1中。 其中有一个机械臂被放在

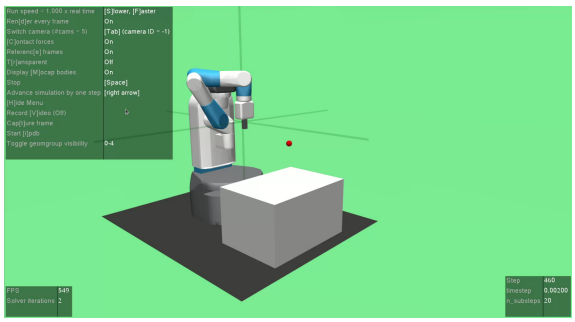

图 3-1 FetchReach-v1 环境的截图

一张桌子前。在每个片段刚开始时，这个机械臂的状态都会被重置，而一个红

点则会随机地出现在桌子上方的某处。机械臂智能体需要在片段的限制时间内使自己的末端执行器到达这个红点附近来完成任务并获得奖励。在片段的每一个时间步中，都会有观测值提供给此机械臂智能体。观测值包括指示关节状态、末端执行器位置坐标信息作为已完成的目标，和红点的位置坐标信息作为期望目标。指示关节状态的观测值 $o \in \mathbb{R}^{10}$ 是一个 10 维的向量，而末端执行器的坐标 $g^a \in \mathbb{R}^3$ 和红点的位置坐标 $g^d \in \mathbb{R}^3$ 都是 3 维的向量。因此提供给机械臂智能体的状态向量是一个 16 维的向量 $s = (o, g^a, g^d) \in \mathbb{R}^{16}$。如果末端执行器和红点之间的距离小于一个阈值，一个值为 0.0 的奖励会被提供给机械臂智能体，否则一个值为-1.0 的奖励会被提供给它。机械臂可以采取的动作是一个 4 维的取值在-1 到 1 之间的向量，即 $a \in [-1, 1]^4$。

3.2 Pyrobolearn 和 Pybullet

虽然使用 Gym 和 Mujoco 的功能已经可以覆盖本课题的大部分实验，但是由于 Mujoco 是商业仿真软件，且可定制性不强，因此有必要使用开源的替代品，即 Pyrobolearn 和 Pybullet，来设计本课题中的主体实验。

Pyrobolearn 是一个专门设计来训练智能机器人的框架[21]。它目前仍然在开发中，但是大部分实验中用到的功能已经足够稳定。一些尚未在其中实现的功能也可以通过继承现有类来进行自定义。与 Gym 相似，Pyrobolearn 也有一个默认的物理引擎。这个物理引擎叫做 Pybullet[31]。它是一个 C++ 编写的开源的物理引擎 Bullet3 在 Python 接口的封装，并可以提供本文中需要的所有仿真功能。它可以实时地检测碰撞，并可以精确地仿真多物理现象，这保证了它可以被用于本文需要的机器人和物体交互仿真的研究需求。虽然 Pyrobolearn 也有一个使用 Mujoco 作为仿真器的接口，但是它对 Mujoco 的支持尚未稳定，因此本文的实验选择了使用 Pybullet 作为仿真器。

使用 Pybullet 作为仿真器，Pyrobolearn 可以被用来仿真多种经典的机器人控制和强化学习问题。实际使用中，一个物理世界对象包含了一些固定的物理量，例如重力方向和摩擦系数，并可以加载物体和机器人。一个世界对象在构造过程中通常要提供一个仿真器，本文提供了 Pybullet 作为仿真器。在世界对象构造完成后一个由仿真器生成的世界相机会被提供给世界对象作为一个属性。

在接下来的使用 Pyrobolearn 的实验中，全部都使用了 *Basic World* 类型的世界对象，其中重力加速度向量为笛卡尔坐标系中的 (0.0, 0.0, -9.81)，横向摩擦系数为 1.0，旋转摩擦系数为 0.0，滚动摩擦系数为 0.0，线性阻尼为 0.04，角阻

尼为 0.04。

渲染后的世界相机下的 *Basic World* 世界如图3-2所示。 默认情况下，没有

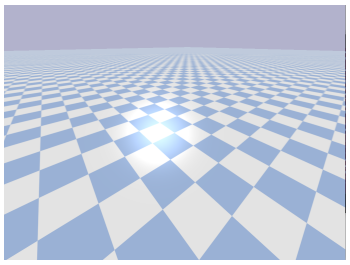

图 3-2 空白的 *Basic World* 世界渲染结果

物体和机器人被加载，只有地面和基本物理量被初始化。

两种不同的机械臂分别在初步实验和主体实验中使用。其中一个来自于 Gym 环境，另一个来自于 Pyrobolearn 框架。

在基于 Mujoco 的几个 Gym 环境中，一个和 Fetch 研究平台中的 Fetch 便携机械臂具有相同参数的机械臂被用作默认的唯一一个与环境交互的机器人[32]。Fetch 便携机械臂有 7 个自由度。在本文中使用到的 *FetchReach-v1* 环境中，只有其中的 4 个被使用。如第 1 节所述，*FetchReach-v1* 环境中的状态向量由 Fetch 机械臂的关节的位置信息、末端执行器的笛卡尔坐标和红点的笛卡尔坐标构成。

Pyrobolearn 框架中的主体实验使用了一个叫做 WAM 的机械臂。WAM 机械臂可以配备手指终端执行器，它们一共有 15 个自由度，这意味着动作空间是非常高维的。在 *Basic World* 世界中加载的 WAM 机器人如图3-3所示。

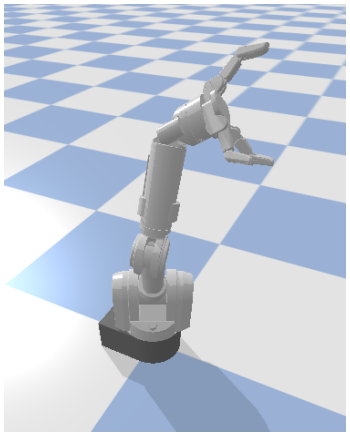

图 3-3 *Basic World* 中的 WAM 机器人渲染结果

3.3 实验框架

本文中所有的实验都有可更改目标的状态，这意味着智能体可以获知当前已完成的目标和期望的目标。假设期望的目标用一个 3 维的笛卡尔坐标表示，

已完成的目标用机器人的末端执行器的笛卡尔坐标表示。通常情况下，当这两个坐标间的距离小于一个阈值时，表示任务已经顺利完成。

形式化地，一个状态向量 $s \in \mathcal{S}$ 是一个由 3 个向量构成向量，即 $s = (o, g^a, g^d)$，其中 o 是观测到的环境中的物理量构成的向量，g^a 是已完成的目标，g^d 是期望的目标。一个环境奖励函数 $reward : \mathcal{S} \rightarrow \mathbb{R}$ 是由环境反馈决定的智能体在交互中可获得的奖励。

在实验中，因为目标可以方便地修改，所以可以使用事后经验重放算法，这也使得定义具有稀疏奖励的开放任务，并在其中使用多种方法训练高探索效率和泛化能力的智能体成为可能。

智能体与环境的交互在实验中被组织为分离的片段。每个片段都有相同的最大时间步数 T。智能体应当在最大时间步数的限制 $t \leq T$ 下完成一个特定的开放任务并获得较高的奖励。对于 *FetchReach-v1* 环境，最大的时间步数为 50. 对于在 Pyrobolearn 框架下的主体实验中自定义的环境中，最大的时间步数为 100 或 200. 智能体可以把在交互中获得的迁移信息 (s_t, a_t, r_t, s_{t+1}) 保存在重放缓冲[33] 中以便之后采样和训练。

对每个片段，最后一个时间步的下一个状态 s_{T+1} 不应被用于训练，因为当环境处理这个状态时这个片段已经结束，因此它不会导致未来的任何奖励。为了防止在训练过程中使用这个状态，可以引入一个掩膜变量 m，并与根据 s_{t+1} 计算得出的动作价值函数值相乘。当 $t < T$ 时，令 $m = 1$，当达到最后一个时间步，即 $t = T$ 时，令 $m = 0$，并将 m 和迁移信息一同放入重放缓冲中以便训练时与上述动作价值函数相乘。

在每个实验的片段开始时都引入了随机初始化[34]。

对于 *FetchReach-v1* 环境中的实验，在片段的第一个时间步之前，Fetch 便携机械臂的关节位置被初始化为一个固定的起始位置，由红点表示的期望目标则随机地初始化为桌面上的一个位置。

在自定义的 Pyrobolearn 环境中，在片段开始之前，WAM 机械臂上的关节位置被随机初始化，期望的目标也被随机地初始化在 1 米以内，其中目标的 z 坐标被初始化为 0.5，x, y 则从 $[-1, 1]$ 的均匀随机分布中采样。与 *FetchReach-v1* 环境中相比，自定义的环境中机械臂的初始状态是随机的，这会给此开放任务带来更多的复杂性。有时候 WAM 机械臂的随机初始状态距离目标非常远，这会导致任务理论上不可能完成。

3.4 Gym 初步实验系统设计

Gym 环境下的实验程序按照类图3-4所示的方式组织。其中 Module 是 Py-

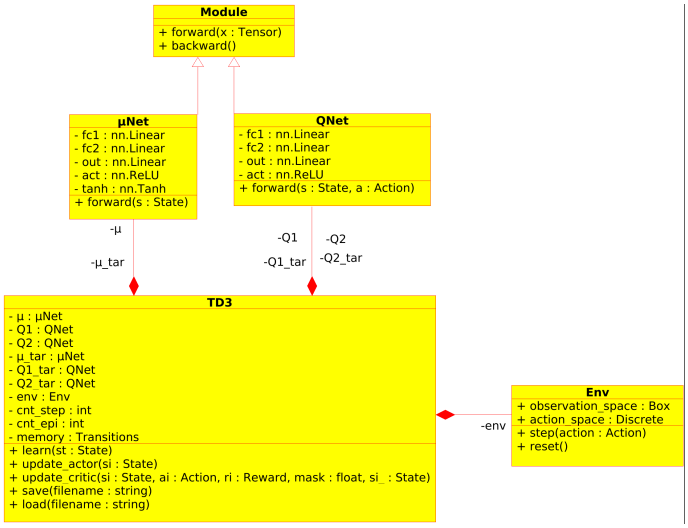

图 3-4 Gym 初步实验程序类图

torch 中的神经网络模块，Net 和 QNet 通过继承自动获得反向传播的方法 backward。μ，Q1 和 Q2 分别是演员网络，评论家网络 1 和评论家网络 2；μ_tar，Q1_tar 和 Q2_tar 分别是对应的靶网络。每调用一次 TD3 对象的 learn 方法，都会进行一个时间步的仿真，时间步计数器 cnt_step 和片断计数器 cnt_epi 相当地进行自增运算，并把生成的迁移 $(s_t, a_t, r_t, mask_t, s_{t+1})$ 存入重放缓冲 memory 中。

上述仿真过程是通过调用环境对象 env 的 step 方法来实现的，step 方法接收根据演员网络在接收当前状态 s_t 后输出的动作结合相应随机策略生成的动作 a_t，进行一个时间步的仿真，并输出下一个状态 s_{t+1}，奖励值 r_t，此片断是否已经结束 *done* 和其他信息。在3.3中所述的掩膜 m 可以与折扣系数 γ 合并，根据如下公式计算新的掩膜：

$$m = \begin{cases} 0, & 若done = True \\ \gamma, & 其他 \end{cases}$$

当 cnt_step 和 cnt_epi 满足一定条件时，update_actor 方法或 update_critic 方法会被触发。在 update_actor 和 update_critic 方法中，首先对重放缓冲 memory 进行采样，采样后对采集到的批次迁移数据根据2.3中描述的过程进行事后经验重放。最后使用如下定义的损失函数计算梯度，使用 backward 方法进行反向传播，并修改网络权重：

$$L_{\mu} = -\sum_{i}^{N} \frac{Q_1(s_i, \mu(s_i))}{N} \tag{3-1}$$

$$L_{Q_1} = \sum_{i}^{N} \frac{r_i + m \min\{Q_1'(s_{i+1}, \mu(s_{i+1})), Q_2'(s_{i+1}, \mu(s_{i+1}))\} - Q_1(s_i, a_i)}{N} \tag{3-2}$$

$$L_{Q_2} = \sum_{i}^{N} \frac{r_i + m \min\{Q_1'(s_{i+1}, \mu(s_{i+1})), Q_2'(s_{i+1}, \mu(s_{i+1}))\} - Q_2(s_i, a_i)}{N} \tag{3-3}$$

其中 $\mu, Q_1, Q_2, \mu', Q_1', Q_2'$ 分别对应前述 μ，Q1，Q2，μ_tar，Q1_tar 和 Q2_tar。为保证实验结果可复现，在经过多次迭代之后要使用 save 方法保存网络权重，并在测试时使用 load 方法加载。

上述 Q1 网络，Q2 网络，Q1_tar 网络和 Q2_tar 网络都具有相同的网络结构，它们的网络结构如图3-5所示。

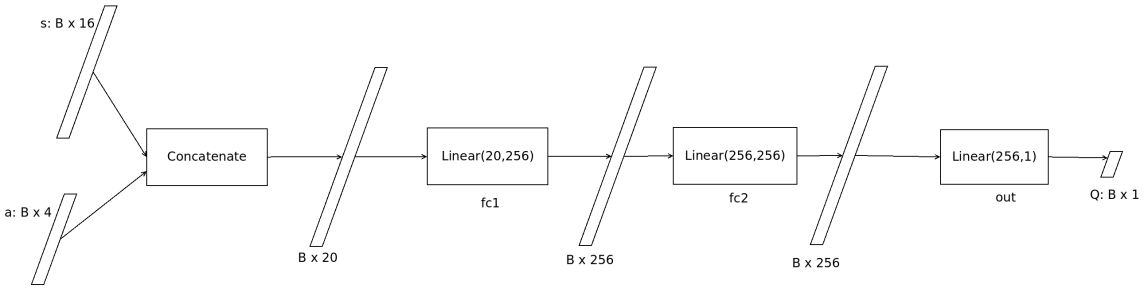

图 3-5 Gym 环境下初步实验的评论家网络结构

上述 μ 网络和 μ_tar 网络具有相同的网络结构，其网络结构如图3-6所示。

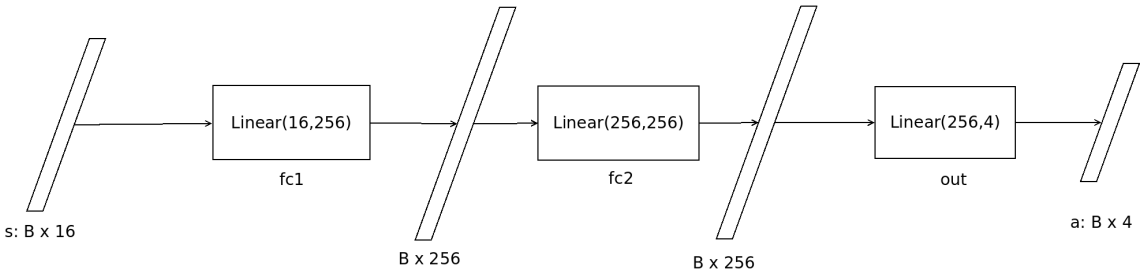

图 3-6 Gym 环境下初步实验的演员网络结构

3.5 Pyrobolearn 主体实验系统设计

在 Pyrobolearn 框架中自定义的环境中有 3 种不同类型的状态构成了最终的状态向量：零件（link）世界速度状态、零件世界位置状态和物体位置状态。在 Pyrobolearn 中，世界中的对象和机器人都被叫做物体（body）。每个物体都有一个唯一的物体 ID，它可以被世界对象用来找到对应的状态。而机器人上面互相连接的构件被叫做零件（link）。每个零件也有一个唯一的零件 ID，它可被机器

人对象用来找到对应的信息，而不能被世界对象直接访问。零件的状态必须使用零件的 ID 来生成，使用物体 ID 生成零件状态或用零件 ID 生成物体状态都会导致严重的隐藏错误。

在实验中，WAM 机械臂的末端执行器的 ID 可以使用 get_end_effector_ids 来得到，而最后一个末端执行器的 ID 将在实验中用来生成已完成的目标向量 g^a。机器人上的一个零件的零件世界位置状态是一个 3 维的向量，而一个零件的零件世界速度状态是一个 6 维的向量。如上所述，在实验中，直接把 WAM 机械臂的最后一个末端执行器的 ID 提供给了零件世界位置状态对象的构造方法，此状态对象可以用于监控已完成的目标向量 g^a 的值。除了上述被选中的最后一个末端执行器，WAM 机械臂还有 20 个其他零件。每一个零件都有一个 3 维的世界位置状态和一个 6 维的世界速度状态。

实验中会有两个物体加载进 *Basic World* 世界中，一个是上述 WAM 机械臂，另一个是一个初始位置随机的蓝色方块。主体实验定义的开放任务为，WAM 机械臂的最后一个末端执行器距离蓝色方块的距离应当在片段的最大时间步数限制内缩小到小于一个阈值，在每一个时间步，如果距离大于此阈值，则提供-1.0 的奖励，否则提供 0.0 的奖励。为了方便使用事后经验重放算法，需要在每个时间步获得蓝色方块的世界位置，用于监控这个期望目标向量 g^d 的状态可以很容易地使用方块的物体 ID 和物体世界位置构造方法来得到。

上文已经介绍了构成状态向量 $s = (o, g^a, g^d)$ 中的已完成的目标 g^a 和期望目标 g^d 的定义，余下的 o 定义为 WAM 机械臂上的除了最后一个末端执行器的其他零件的世界位置和世界速度。最终，这些状态构成了一个 192 维的状态向量 $s \in \mathbb{R}^{192}$。主体实验中使用到的这些状态并不是所有的状态。根据通用机器人描述格式（Universal Robotic Description Format[35]）中描述的模型，还有其他的状态（如图像传感器状态）可以被添加，但是现有的状态信息已经足够多以使得机械臂能够完成此开放任务，因此实验中没有添加更多状态。

在主体实验中，只使用了关节位置变化类型的动作作为机械臂与环境交互的动作。在机械臂做出一个关节位置变化动作之后，关节的位置与动作值相加即可得到新的关节位置。关节位置动作是一个 15 维的向量，这与 *FetchReach-v1* 环境中的动作维度要高得多，因此在 Pyrobolearn 中自定义的此开放任务要比前者更难。

所有的实验中用到的动作和状态都是连续的，当考虑到状态空间和动作空间都是连续的这一事实后，可以知道智能体要搜索的空间是非常巨大的，这与上文定义的稀疏奖励结合起来就引出了在此开放任务下探索策略的问题。在接

下来的实验中，多个探索策略被提出以获得更高效率的探索，并更好地解决此问题。

Pyrobolearn 下的实验程序的类图如图3-7所示。其与 Gym 环境中的初步实

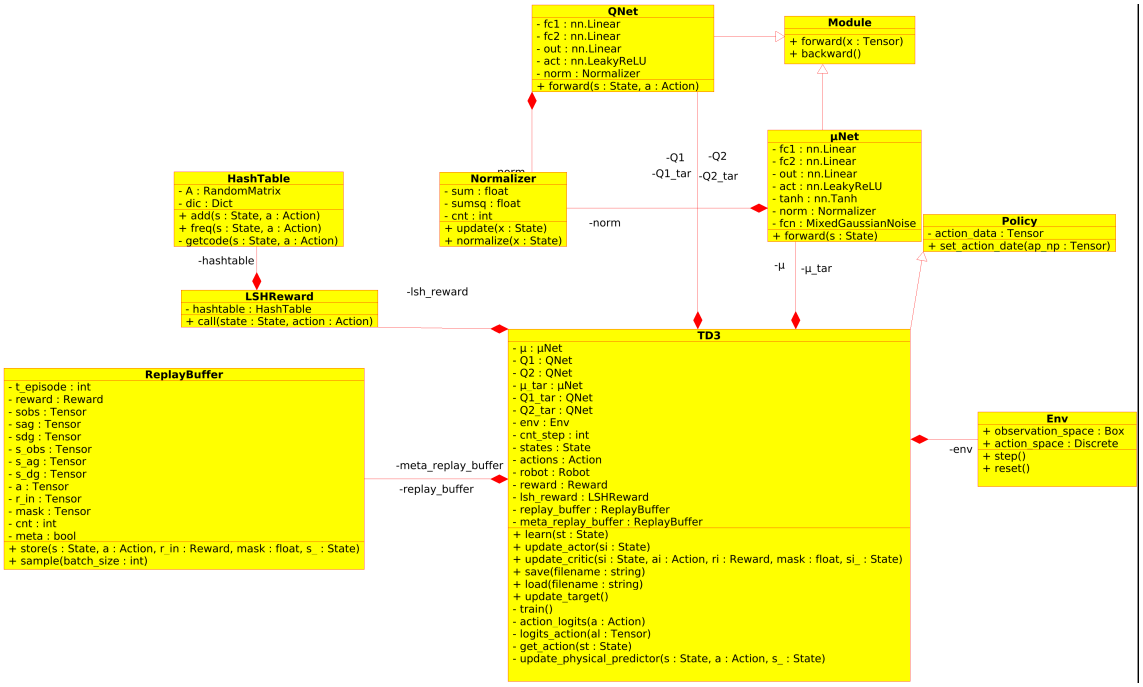

图 3-7 Pyrobolearn 主体实验程序类图

验程序的主要区别在于引入了基于局部敏感哈希的内部奖励函数对象 LSHReward，正态化器 Normalizer[36]，Leaky ReLU 激活层[37]，高斯噪声层 MixedGaussianNoise 和元重放缓冲 meta_replay_buffer 的引入。元重放缓冲用于在 $t < T_{start}$ 的时间内，未提供环境奖励时，保存内部奖励对应的迁移。另外在其中一个主体实验中还使用了 update_physical_predictor 方法来对正向动力学预测模型进行更新和训练，其中此模型的损失被用于构成内部奖励。为了方便演员网络使用 Tanh 激活层[38] 输出，策略 TD3 中还引入了 action_logits 来把动作向量的每个元素的值都缩放到 [-1,1]，并使用 logits_action 来进行反向解码出动作。

与初步实验中类似，主体实验也使用全连接网络[39,40]，但在演员网络中最后一层加入了混合高斯噪声层并与原网络输出动作求和，并且对输入层输出层进行了相应调整以适应输入数据。由于输入输出主体实验中所有神经网络的动作向量都进行了缩放，演员网络的最后一层不需要再乘以 max_action，只需要直接输出 [-1,1] 的缩放后的动作值，在策略 TD3 中调用演员网络的地方会自动进行缩放。

主体实验中使用的演员网络结构如图3-8所示。主体实验中使用的评论家网络结构如图3-9所示。

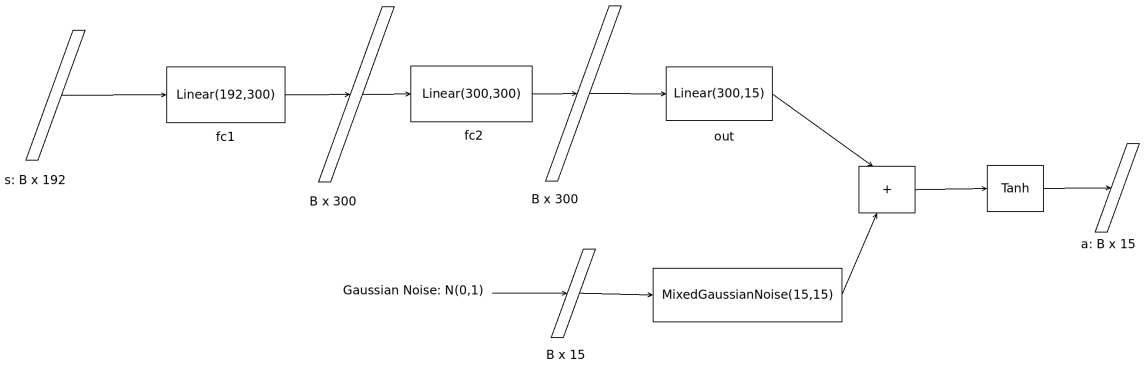

图 3-8 Pyrobolearn 中主体实验演员网络结构

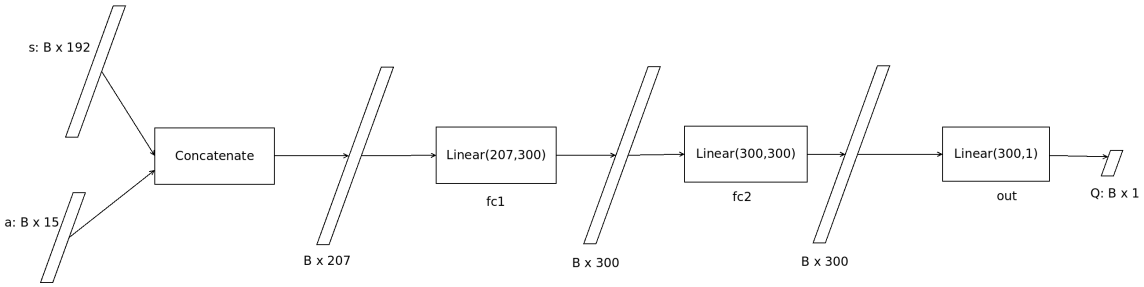

图 3-9 Pyrobolearn 中主体实验评论家网络结构

第 4 章 轨迹替换方法

4.1 工作原理

在 Gym 环境中进行初步实验时，与2.3中的替换方法不同，不在采样时对已完成的目标进行替换，而是在将新的迁移保存到重放缓冲中时对整个片段进行进行替换，并把未替换和替换后的轨迹都加入到重放缓冲中。换句话说，每当一个片段结束时，对这个片段中由所有迁移构成的整个轨迹中的已完成的目标进行替换，并把替换后的一整个新轨迹添加到原轨迹之后。

形式化地，轨迹替换会从重放缓冲中取出一整个轨迹的副本 $s[1:T] = (o[1:T], g^a[1:T], g^d[1:T])$，并对其做如下替换：

$$g^d[1:T] = g^a[-1]$$

$$r[1:T] = reward(g^d[1:T], g^a[1:T])$$

并将所替换后的轨迹添加到重放缓冲中。

实验在3.4中设计的系统中进行，使用的实验参数见表4-1。其中 ϵ_{noise} 是在更

表 4-1 初步实验参数

实验参数	值
ϵ_{noise}	0.2
σ_{clip}	0.5
M	5120
ϵ_{rand}	0.3
γ	0.99
α	3×10^{-4}
B	8
τ	0.005
T	50
f_μ	2
T_{start}	2.5×10^4
N_{sample}	10

新评论家网络权重，使用靶演员网络计算动作向量时添加进去的高斯噪声的标准差，σ_{clip} 是对此噪声进行裁剪的阈值，裁剪后有噪声大小在区间 $[-\sigma_{clip}, \sigma_{clip}]$ 以内。M 是重放缓冲能保存的迁移的最多数量。ϵ_{rand} 是采用 ϵ-greedy 探索方法进行探索时，随机采样动作的概率大小。γ 是前述折扣系数。α 是演员网络和评

论家网络的优化器的学习率。B 是从重放缓冲中采样迁移的批次大小。τ 是2.2中所述的指数滑动平均系数。T 是每个片段的最大时间步数。每经过 f_μ 个时间步，才对演员网络权重进行更新一次，而评论家网络每权重更新和对靶网络权重的指数滑动平均更新在每个时间步都要做。在 $t < T_{start}$ 时，使用完全随机的采样来得到智能体要做的动作，在 $t \geq T_{start}$ 时，才开始使用演员网络获取动作向量。每经过 N_{sample} 个片段记为一代（epoch），对损失、奖励和成功率进行统计，最后绘制关于代的曲线图。

4.2 实验结果

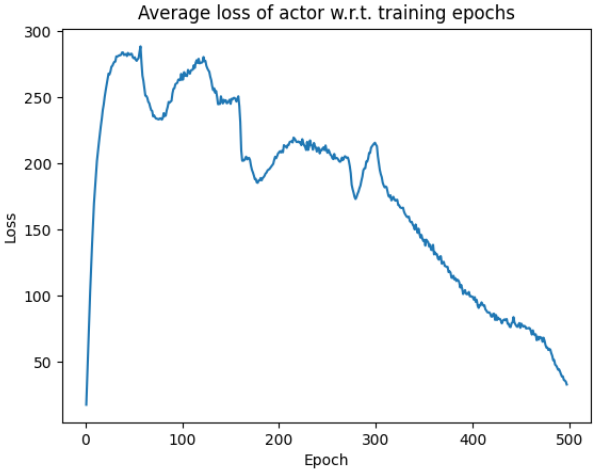

图 4-1 μ 的损失曲线

如图4-1所示，演员网络在每个片段的平均损失在前 60 代快速上升，并在之后有回跳地下降。

评论家网络 1 和评论家网络 2 的每个片段的平均损失如图4-2和4-3。通过对比可以看出两个图中的损失曲线的变化趋势大致相同且剧烈变化之处与演员网络的损失相似，这可能是由于评论家网络的好坏通过损失影响了演员网络表示的确定性策略的好坏。

每一代（10 个片段）的平均环境奖励如图4-4所示。可以看出，随着训练的片段数和代数增多，环境的奖励开始逐渐增大，最终达到-5 左右。

平均每个片段的成功率如图4-5所示。每个片段的成功与否是通过片段末尾的奖励来判断的，若片段末尾的状态中的已完成的目标与期望目标的距离小于给定值，得到了值为 0.0 的奖励，就说这个片段成功完成了任务。由于成功率仅仅是 10 个片段的平均值得到的，因此它只能取到 10 个离散的值。因为成功率

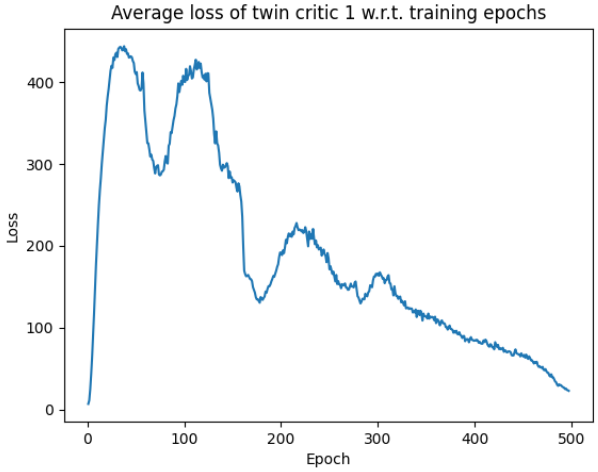

图 4-2 Q_1 的损失曲线

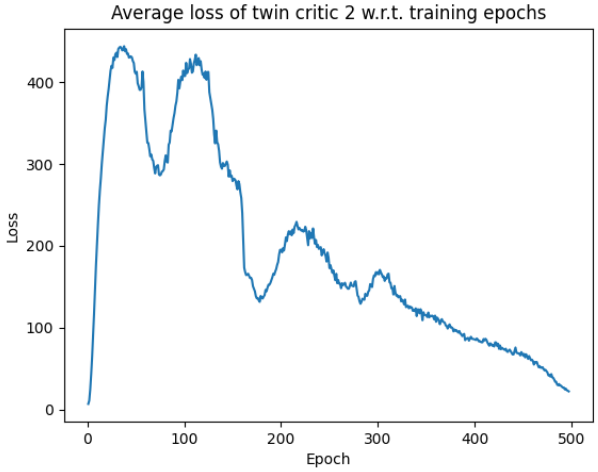

图 4-3 Q_2 的损失曲线

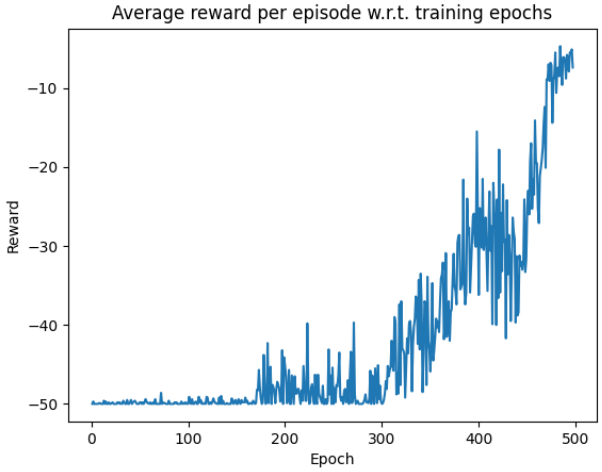

图 4-4 环境奖励

的计算只考虑了最后一个状态，所以成功率曲线与平均奖励曲线不完全相同。

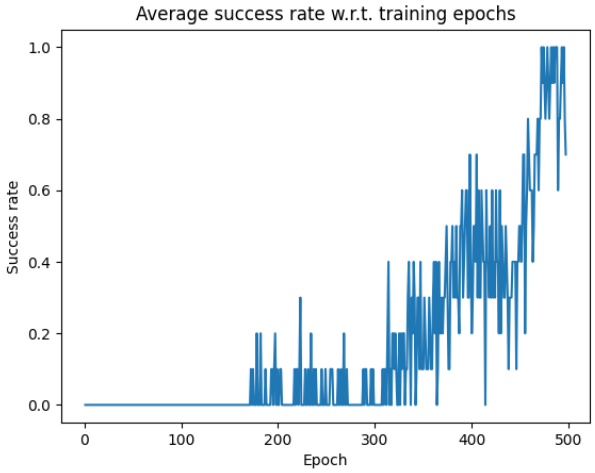

图 4-5 任务目标的平均成功率

在训练完成后，仿真环境 *FetchReach-v1* 中的这个机械臂已经可以控制它的关节并快速地使末端执行器达到红点位置，成功率约为 90%。如图4-6所示，机械臂成功地完成了任务.

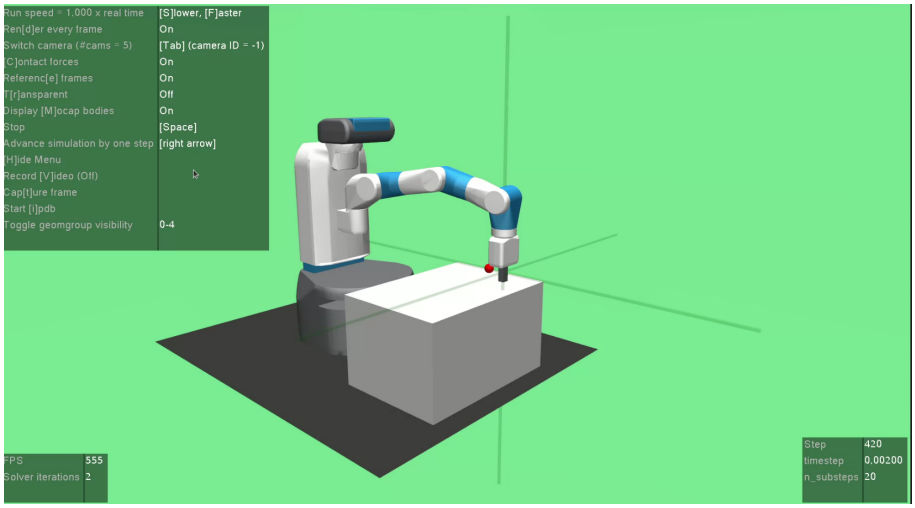

图 4-6 Fetch 便携机械臂控制关节使末端执行器达到了红点位置，成功完成了指定任务目标。

第 5 章 混合高斯噪声层

5.1 工作原理

通常情况下，为了增加策略的稳健性，会对演员网络输出的动作向量添加一个均值为 0，标准差为定值的高斯噪声。得到的动作常常还要进行裁剪以防止出现不合理的动作。然而，此种方法需要对两个超参数进行调节，需要大量的时间来进行训练，寻找最优的超参数值。由于在此主体实验中，往往需要训练很长的时间，因此引入这两个超参数并不理想。

为了避免手动调节噪声大小，本文提出了一个嵌入演员网络的可自动调节的混合高斯噪声。这个自动调节的噪声向量是通过直接在正态噪声向量后添加全连接层来得到的。得到的噪声向量被加到演员网络最后一层的输出上。形式化地，给定一个从正态分布中随机采样的 $B \times \dim(a)$ 的高斯噪声矩阵 $X_{B\times\dim(a)}$，其中 B 是训练的迁移批次大小，$\dim(a)$ 是动作空间的维度大小。如果把演员网络中确定性的部分的输出表示为 $f(s)$，其中 s 是输入的状态向量，则有如下缩放后的噪声：

$$\mu(s) = f(s) + WX + b, X_{ij} \sim \mathcal{N}(0, 1),$$

其中 W 和 b 是全连接层的权重，它们通过反向传播自动调节。

5.2 实验设计

为了更好地展示添加此混合高斯噪声层的效果，研究中在 Pyrobolearn 中的自定义环境中进行了两个实验以进行对照。其中一个有着固定的噪声大小作为超参数，而另一个没有此超参数而是换成了本文提出的混合高斯噪声层。实验系统的基本设计已经在3.5中进行了详细讨论。本实验中使用了2.2中描述的未来替换方法，与3.5中不同的是，在这两个实验中对每个网络都多添加了一层隐藏层。此外，随机初始化时对蓝色方块使用了 $x, y \in [-1, 1]$ 的均匀采样，WAM 机械臂的最后一个末端执行器与方块的距离达到 0.5 米以内即可获得 0.0 的奖励，其余情况获得-1.0 的奖励，且对靶网络的更新每 2 个片段进行一次。使用的实验参数如表5-1所示，其中符号的意义与4-1中相同。

表 5-1 混合高斯噪声层实验参数

实验参数	值
ϵ_{noise}	0.2
σ_{clip}	0.5
M	5×10^3
ϵ_{rand}	0.3
γ	0.991
α	3×10^{-4}
B	16
τ	0.005
T	50
f_μ	2
T_{start}	2.5×10^4
N_{sample}	200
k	18
K_{replay}	4
ξ_{LSH}	2×10^{-2}

其中 k 是2.4中的超参数，用于调整对状态空间离散化的粒度。K_{replay} 是2.3中所述的超参数，它与替换掉的迁移的比例有关。ξ_{LSH} 是在计算最终奖励时，基于局部敏感哈希和计数的探索奖励的系数。其余参数的意义与表4-1中相同。

5.3 实验结果与分析

平均每个片段的使用混合高斯噪声层的演员网络的损失和使用噪声大小超参数的演员网络的损失如图5-1所示。有混合高斯噪声层的演员网络的损失在较少的代数内上升到了一个相对较低的值。由于这两组实验中损失都收敛到了较高的值，此损失曲线并不能明确反映哪种方法更优，因为演员网络的损失是由评论家网络决定的，而这两组实验中的评论家并不一定具有相同权重，因此即使是相同的演员网络也有可能具有不同的损失。

每个片段的评论家网络 Q_1 和 Q_2 的平均损失分别如图5-2和图5-3所示。可以看出，它们的损失也是和演员网络 μ 的损失具有相似的形状。由于评论家网络的损失也并没有收敛到较低的值，这些损失曲线也不能反映混合高斯噪声层带来的变化。但是可以肯定的是，添加混合高斯噪声层后损失曲线变化更加平缓了。从图中也可以看出，评论家网络的损失比演员网络的损失更加不稳定，这可能是由于评论家网络的损失直接与环境相关导致的。

使用和不使用混合高斯噪声层的单位片段平均环境奖励如图5-4所示。可以看出，添加了混合高斯噪声层的方法获得的奖励要稍微比只使用噪声大小超参

图 5-1 演员网络 μ 的损失曲线

图 5-2 评论家网络 Q_1 的损失曲线

图 5-3 评论家网络 Q_2 的损失曲线

数的方法更高。

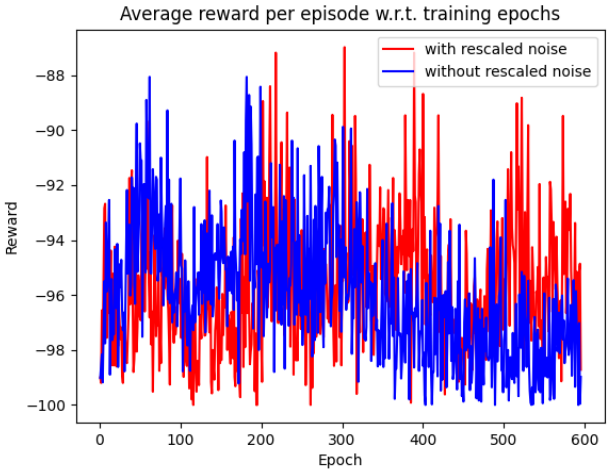

图 5-4 平均环境奖励曲线

平均成功率曲线的变化趋势与平均环境奖励的大致相同。对于最后的平均成功率，添加了混合高斯噪声层的要稍微更高。

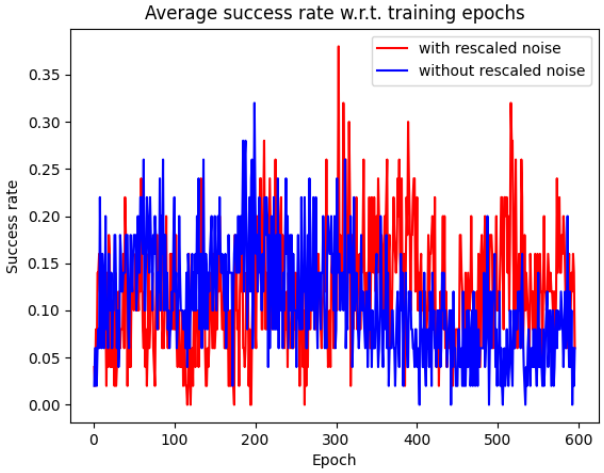

图 5-5 任务目标成功率

第 6 章 仿真时间奖励

6.1 工作原理

在对物理过程进行仿真的过程中，物理引擎往往会做一些缓存和优化来对仿真进行加速，而这种加速是基于此物理过程容易预测这一事实的。这意味着，对于难以进行预测的复杂物理行为，如复杂的碰撞，物理引擎要消耗更多的时间来进行仿真。这就使得利用此信息来为智能体提供任务无关的探索奖励成为可能。

在实验过程中，为了准确计算仿真时间，避免其他因素影响，必须保证在训练过程中没有其他程序在运行。这一要求可以简单地通过避免在训练时操作计算机来满足。此外，为了防止系统调用的时间被计入，在实验中对仿真时间进行计算时，直接在仿真代码被调用前后使用 Python 的 time 库进行了计时。

除了仿真时间奖励外，实验中还提供了基于局部敏感哈希的探索奖励。为了平衡不同的内部奖励，需要在计算最终奖励前先乘上一个系数。

在演员网络中，使用了上一章提出的混合高斯噪声层，但不同的是，为了防止噪声产生过大影响，继续使用参数 σ_{clip} 对噪声进行裁剪。

为了防止机械臂在可以完成任务的动作序列中选择较长的序列，在对演员网络进行优化时，可以在其损失函数中添加一个动作惩罚项，即

$$L_{\mu} = -\sum_{i}^{N} \frac{Q_1(s_i, \mu(s_i))}{N} + \xi_{action} ||\mu(s_i)||^2$$

其中 ξ_{action} 是动作惩罚系数。

在实验中，整个系统都采用3.5中的设计，但是除了使用3.5中的浅层网络结构外，还增加了一个为每个网络添加一个隐藏层的深层网络作为对照组。

由于在使用正向动力学预测模型来计算奖励时，会导致训练速度极大下降，而且在实验中并未发现它能够对算法结果产生有利影响，因此在本章实验中不再展示相关的实验结果和数据。

实验相关的参数见表6-1。其中 ξ_{action} 是动作惩罚系数，$N_{batches}$ 指每次触发对演员网络或评论家网络的训练时迭代的次数。每经过 f_{target} 个时间步，使用指数滑动平均对靶网络进行权重更新。每经过 f_{actor} 个时间步，使用反向传

表 6-1 仿真时间奖励实验参数

实验参数	值
σ_{clip}	0.5
M	300
ϵ_{rand}	0.3
ξ_{action}	1×10^{-3}
γ	0.998
α	1×10^{-3}
B	10000
τ	0.23
T	200
$N_{batches}$	1
f_{target}	200
f_{actor}	400
f_{critic}	200
T_{start}	1×10^{5}
N_{sample}	50
k	18
K_{replay}	4
ξ_{LSH}	2×10^{-2}
ξ_{sim}	10

播对演员网络权重进行更新。每经过 f_{critic} 个时间步，使用优化器对评论家网络权重进行更新。ξ_{sim} 是在计算最终奖励时，仿真时间奖励的系数。其余的参数意义与表5-1中相同。

6.2 实验结果与分析

只有 3 层与有 4 层的网络相比，演员网络损失曲线如图6-1所示。可以看出浅层网络更快地收敛了，而深层网络收敛到了一个损失较大的值。

对于评论家网络来说，如图6-2和6-3所示，情况是类似的，更加值得注意的是，深层网络更容易出现损失突然暴增的现象，这意味着深层网络训练起来更加不稳定。

浅层网络和深层网络的平均环境奖励如图6-4所示，可以看出浅层最后学习到了一个较好的策略，得到的奖励显著比深层网络更高，而深层网络的奖励一直维持在最差的结果附近。

观察图6-5中所示的基于局部敏感哈希的计数奖励可以发现，浅层网络的基于局部敏感哈希的计数奖励与深层网络的相差不多，甚至在 200 代附近深层网络反而获得了更高的奖励。

而仿真时间奖励则能比基于局部敏感哈希的计数奖励更好地反映策略的

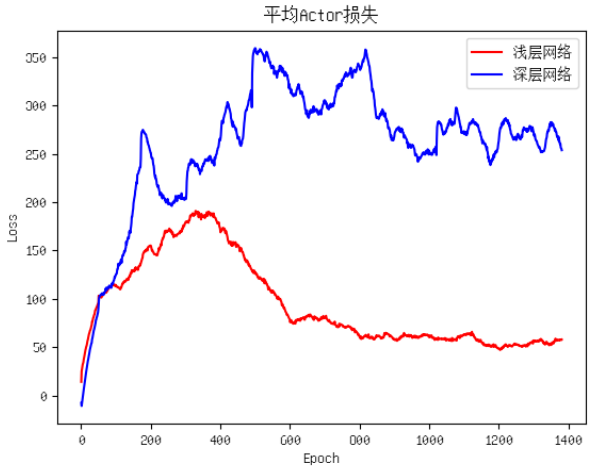

图 6-1 3 层和 4 层演员网络 μ 的损失曲线

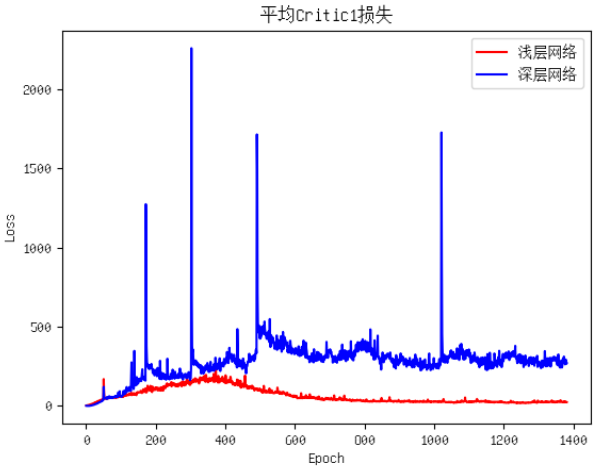

图 6-2 3 层和 4 层评论家网络 Q_1 的损失曲线

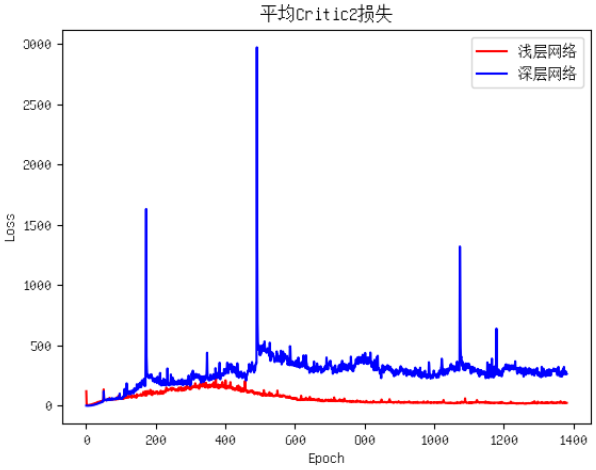

图 6-3 3 层和 4 层评论家网络 Q_2 的损失曲线

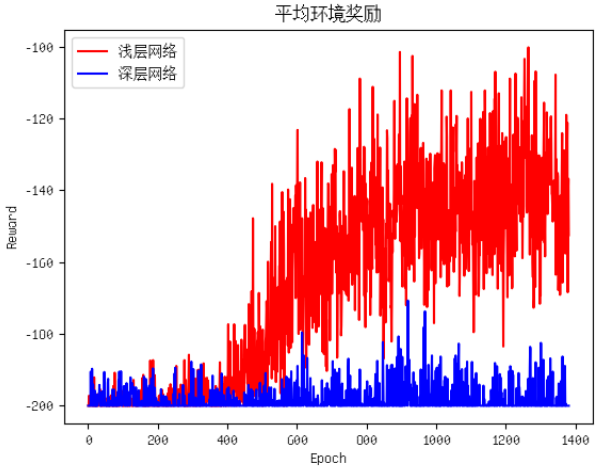

图 6-4 浅层神经网络和深层神经网络的平均环境奖励曲线

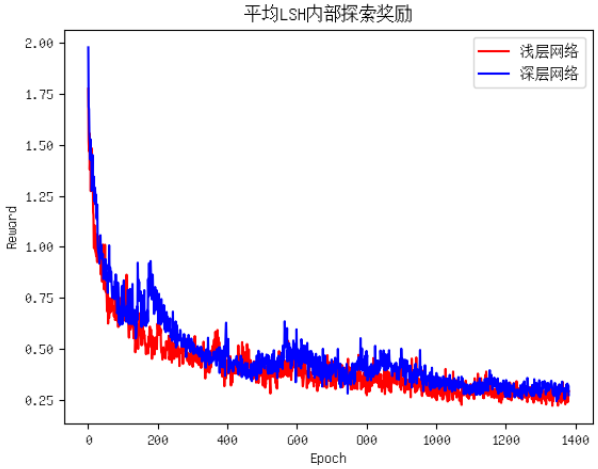

图 6-5 浅层网络和深层网络的基于局部敏感哈希的平均计数奖励曲线

好坏，如图6-6所示，仿真时间奖励在 900 代之后浅层网络要比深层网络更高，这与环境奖励的趋势吻合，表明仿真时间奖励不仅具有任务无关的鼓励智能体探索的能力，还能帮助智能体在探索之后策略泛化到给定开放任务中。

图6-7中的成功率变化与环境奖励的变化趋势相似，这和在其他实验中观察到的一致。

在对深层网络的梯度进行观测时，可以观察到演员网络中的梯度快速地减小至 0，这意味着发生了梯度消失，如果使用残差网络、层归一化或进行谱归一化，则有可能进一步提高性能。

在训练完成后，智能体完成接近方块的开放任务的成功率约为 60%，如图6-8所示为智能体成功完成任务时的截图。

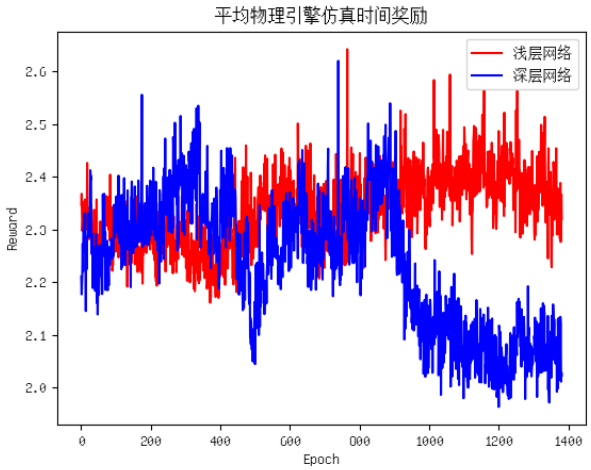

图 6-6 浅层网络和深层网络的平均仿真时间奖励曲线

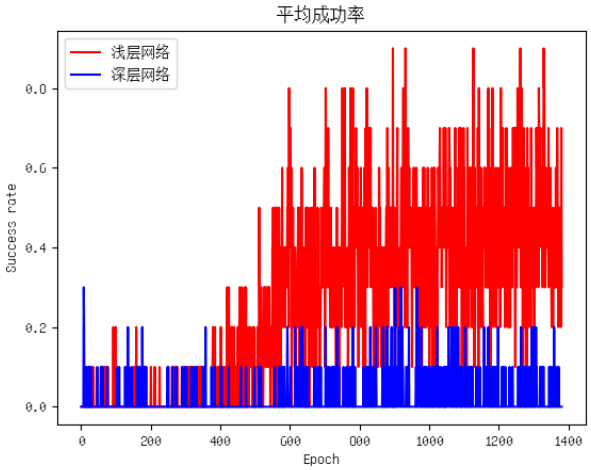

图 6-7 浅层网络和深层网络平均任务目标成功率曲线

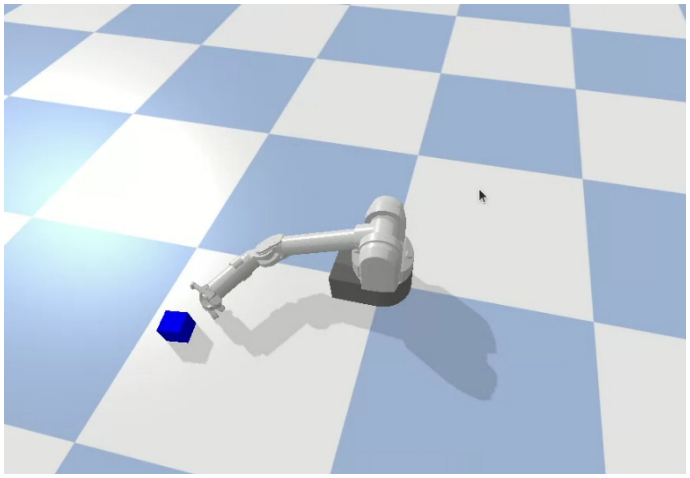

图 6-8 机械臂智能体完成任务的结果展示

第 7 章 对比实验

在之前的章节中，对本文提出的轨迹替换、混合高斯噪声层和仿真时间奖励等方法进行了初步实验验证。本章将通过严格控制变量的对比实验来进一步研究提出部分方法的有效性。

7.1 混合高斯噪声层对比实验

在混合高斯噪声层的对比实验中，除了在实验组的演员网络 μ 添加了混合高斯噪声层以外，其余的算法参数均被设置为相同的数值。详细实验参数见表7-1。其中各个实验参数的符号含义与上一章中的参数表6-1含义相同。

表 7-1 混合高斯噪声层对比实验中的实验组与对照组

实验参数	实验组	对照组
σ_{clip}	0.5	0.5
M	300	300
ϵ_{rand}	0.3	0.3
ξ_{action}	1×10^{-3}	1×10^{-3}
γ	0.998	0.998
α	1×10^{-3}	1×10^{-3}
B	10000	10000
τ	0.23	0.23
T	200	200
$N_{batches}$	1	1
f_{target}	200	200
f_{actor}	400	400
f_{critic}	200	200
T_{start}	1×10^{5}	1×10^{5}
N_{sample}	10	10
k	18	18
K_{replay}	4	4
ξ_{LSH}	2×10^{-2}	2×10^{-2}
ξ_{sim}	0	0
是否使用混合高斯噪声层	是	否

添加了混合高斯噪声层的实验组与未添加混合高斯噪声层的对照组的演员网络损失曲线如图7-1所示。观察损失曲线可以发现，表示实验组的红色曲线上升到了更低的值并下降到了更低的值，这表明混合高斯噪声层可以增加训练过程的稳定性，减少损失发散的可能性。

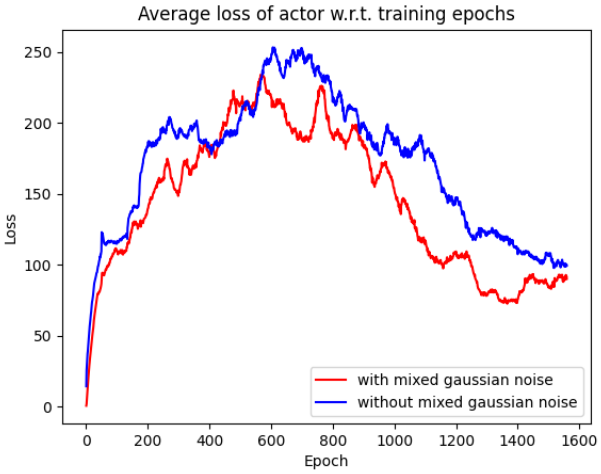

图 7-1 实验组与对照组演员网络 μ 的损失曲线，其中红色曲线为添加了混合高斯噪声的实验组，蓝色曲线为未添加混合高斯噪声层的对照组

对于添加混合高斯噪声层的实验组和对照组的评论家网络的损失，如图7-2和7-3所示，变化趋势与演员网络类似。从图中可以看出，添加了混合高斯噪声层的实验组评论家网络损失没有像对照组那样出现大量尖峰，这意味着实验组的训练过程要比对照组更稳定。与此同时，实验组的两个评论家网络的损失大部分时间都要比对照组的损失略低，这也像演员网络损失曲线一样表明添加混合高斯噪声层的实验组智能体更不容易在训练过程中发散。

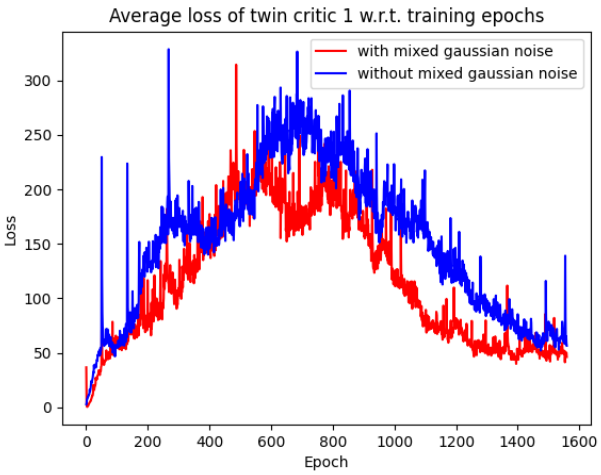

图 7-2 实验组与对照组评论家网络 Q_1 的损失曲线，其中红色曲线为添加了混合高斯噪声层的实验组，蓝色曲线为未添加混合高斯噪声层的对照组

图7-4是添加了混合高斯噪声层的实验组和对照组在一代（10 个片段）内的平均环境奖励。观察两个曲线可以发现，实验组在 1200 代时已经收敛到了比较高的平均环境奖励值，而对照组的平均环境奖励上升更慢且比实验组的奖励值更低。这表明添加了混合高斯噪声层后，智能体在环境中探索并获得奖励的能

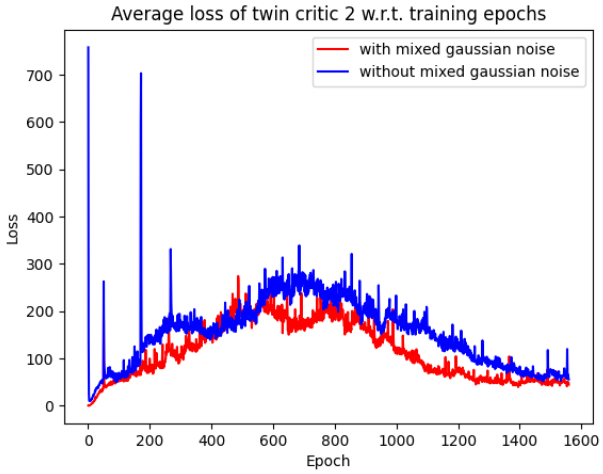

图 7-3 实验组与对照组评论家网络 Q_2 的损失曲线，其中红色曲线为添加了混合高斯噪声层的实验组，蓝色曲线为未添加混合高斯噪声层的对照组

力获得了显著的提升，这也意味着混合高斯噪声层确实能帮助智能体得到更强的探索能力。图7-5中的平均任务目标成功率也显示了和平均环境奖励曲线相似的形状。这表明混合高斯噪声层确实能增强智能体成功完成目标任务的能力。

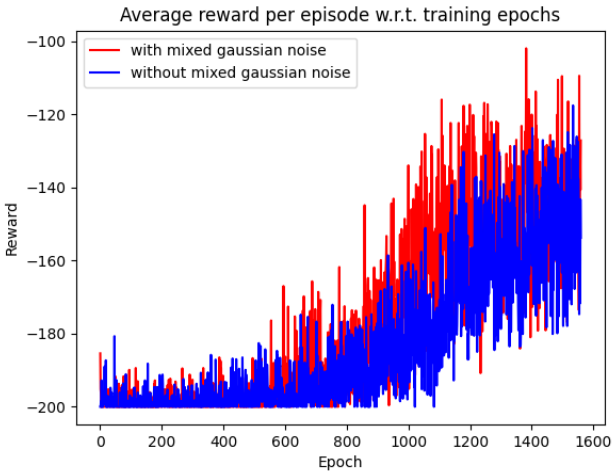

图 7-4 实验组和对照组的平均环境奖励曲线，其中红色曲线为添加了混合高斯噪声层的实验组，蓝色曲线为未添加混合高斯噪声层的对照组

两组实验的基于局部敏感哈希的平均计数奖励曲线如图7-6所示。图中表示实验组的红色曲线在大部分时间都位于表示对照组的蓝色曲线上方。而基于局部敏感哈希的计数奖励表示了智能体探索到更不常见状态的多少，这表明添加混合高斯噪声层后，智能体更经常访问不常见的状态，从而导致了更多的奖励。因此添加了混合高斯噪声层可以改善智能体的探索策略，使之更偏向于访问更新奇的状态。

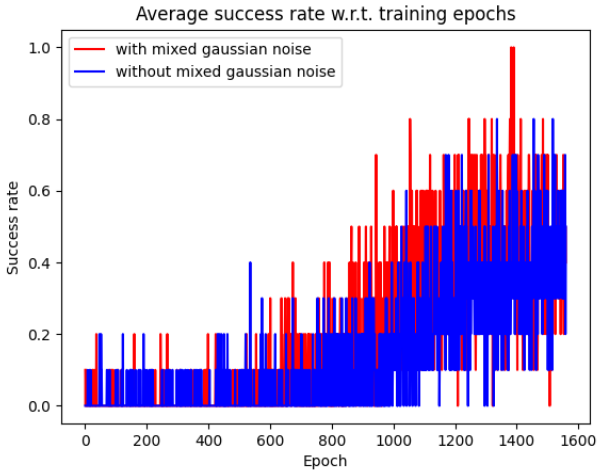

图 7-5 实验组和对照组的平均任务目标成功率曲线，其中红色曲线为添加了混合高斯噪声层的实验组，蓝色曲线为未添加混合高斯噪声层的对照组

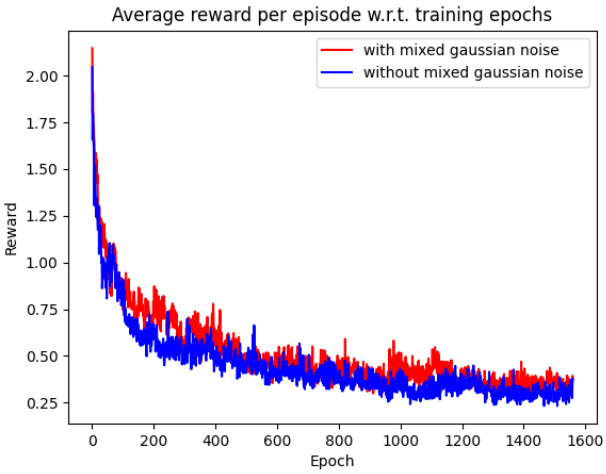

图 7-6 实验组和对照组的基于局部敏感哈希的平均计数奖励曲线，其中红色曲线为添加了混合高斯噪声层的实验组，蓝色曲线为未添加混合高斯噪声层的对照组

7.2 内部奖励的对比实验

在上一章中的实验里，既使用了基于局部敏感哈希的计数奖励，又使用了仿真时间奖励，因此无法对比两种方法。因此在本节的对比实验中，除了实验组使用的内部奖励为仿真时间奖励，对照组使用的内部奖励为基于局部敏感哈希的计数奖励外，其余参数均保持相同。详细实验参数见表7-2。使用基于局部敏感哈希的计数奖励的对照组和使用仿真时间奖励的实验组的演员网络损失曲线如图7-7所示。可以看出表示实验组的蓝色曲线更快地下降到较低的损失值，这表示使用仿真时间奖励不会像基于局部敏感哈希的计数奖励那样由于奖励随时间变化更导致收敛速度变慢。

表 7-2 内部奖励对比实验中的实验组与对照组

实验参数	实验组	对照组
σ_{clip}	0.5	0.5
M	300	300
ϵ_{rand}	0.3	0.3
ξ_{action}	1×10^{-3}	1×10^{-3}
γ	0.998	0.998
α	1×10^{-3}	1×10^{-3}
B	10000	10000
τ	0.23	0.23
T	200	200
$N_{batches}$	1	1
f_{target}	200	200
f_{actor}	400	400
f_{critic}	200	200
T_{start}	1×10^5	1×10^5
N_{sample}	10	10
k	18	18
K_{replay}	4	4
ξ_{LSH}	0	2×10^{-2}
ξ_{sim}	10	0
是否使用混合高斯噪声层	是	是

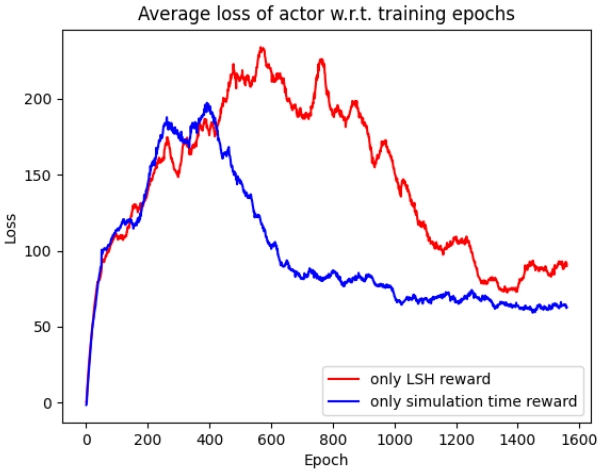

图 7-7 实验组与对照组演员网络 μ 的损失曲线，其中红色曲线为使用基于局部敏感哈希的计数奖励的对照组，蓝色曲线为使用仿真时间奖励的实验组

如图7-8和图7-9所示是实验组和对照组的评论家网络损失曲线。在两个图中除了训练刚开始时以外，都没有较为剧烈的尖峰，这表明改变内部奖励不会引起训练的不稳定性变化。此外，蓝色曲线也和演员网络损失曲线中一样更快地下降，这表明仿真时间奖励的训练过程要更快。

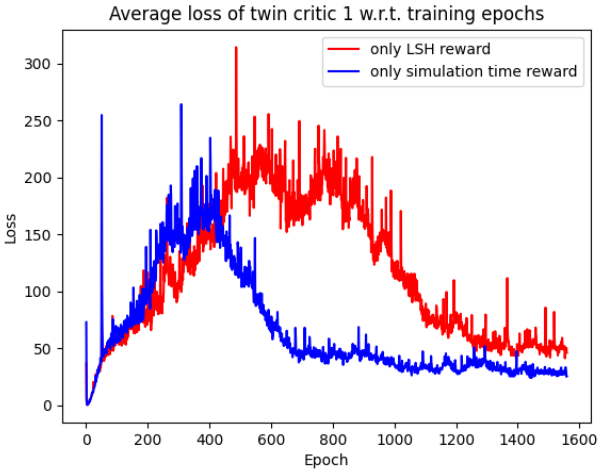

图 7-8 实验组与对照组评论家网络 Q_1 的损失曲线，其中红色曲线为使用基于局部敏感哈希的计数奖励的对照组，蓝色曲线为使用仿真时间奖励的实验组

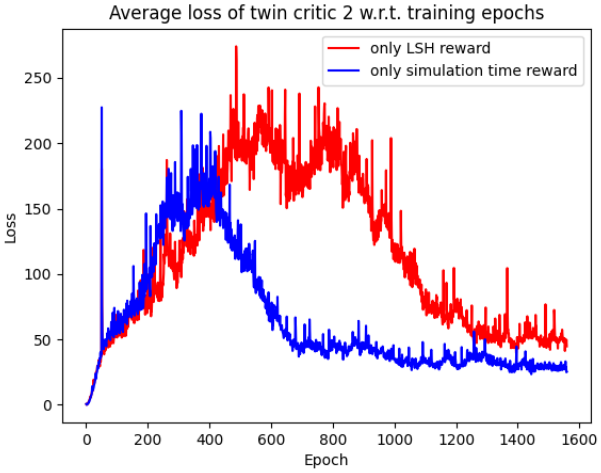

图 7-9 实验组与对照组评论家网络 Q_2 的损失曲线，其中红色曲线为使用基于局部敏感哈希的计数奖励的对照组，蓝色曲线为使用仿真时间奖励的实验组

图7-10和图7-11分别是平均环境奖励曲线和平均任务目标成功率曲线。在两个图中，蓝色曲线表示的仿真时间奖励都更快地收敛。而且大部分时间下，使用仿真时间奖励的智能体都获得了更高的环境奖励，达到了更高的任务成功率。这表明仿真时间奖励要比基于局部敏感哈希的计数奖励能更有效地帮助智能体在环境中探索并在之后的开放任务中取得更好的成绩。

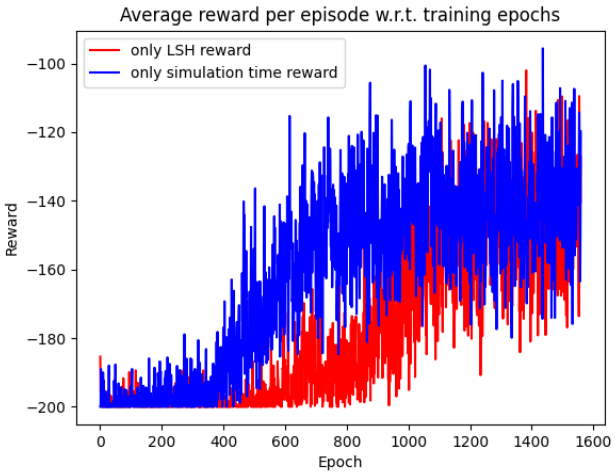

图 7-10 实验组和对照组的平均环境奖励曲线，其中红色曲线为使用基于局部敏感哈希的计数奖励的对照组，蓝色曲线为使用仿真时间奖励的实验组

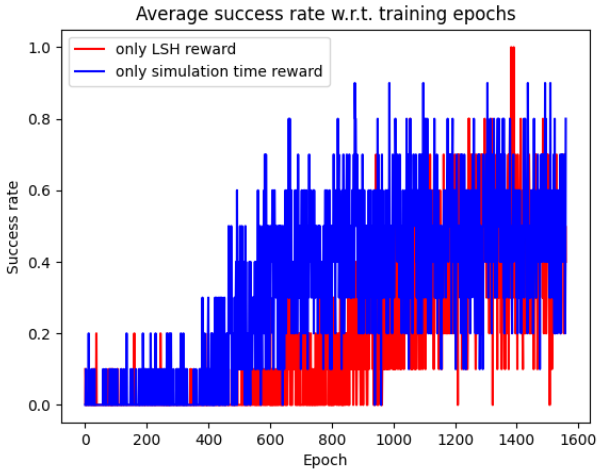

图 7-11 实验组和对照组的平均任务目标成功率曲线，其中红色曲线为使用基于局部敏感哈希的计数奖励的对照组，蓝色曲线为使用仿真时间奖励的实验组

结　论

本文设计了一个控制机械臂接触方块的开放任务环境，并在基于此环境设计了各种算法用于解决智能体在开放环境中探索的问题。

本文提出的算法结合了TD3算法、事后经验重放算法和基于局部敏感哈希的计数奖励，并提出了轨迹替换方法、混合高斯噪声层和将物理仿真时间作为于开放任务的探索奖励。

本文中的轨迹替换方法通过将原有片段的经验修改为成功的片段经验，有效解决了开放任务中的稀疏奖励问题，且与未来替换方法相比，可适用于更多更复杂的场景和任务。

本文提出的混合高斯噪声层实际上将原有的 TD3 算法从确定性策略梯度算法改为了非确定性算法。通过包含可以通过反向传播学习到的混合高斯噪声，智能体获得了更好的探索策略，获得了更高的平均环境奖励和任务成功率。

本文中的物理仿真时间探索奖励可以很好地用于在仿真环境中训练可适应到给定开放任务的智能体，表现出比基于局部敏感哈希的计数奖励更好的探索性能。

参考文献

[1] Silver D, Hubert T, Schrittwieser J, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play.[J]. Science, 2018, 362(6419) : 1140-1144.

[2] Dosovitskiy A, Ros G, Codevilla F, et al. CARLA: An Open Urban Driving Simulator[J]. Conference on Robot Learning, 2017 : 1-16.

[3] Toussaint M, Allen K R, Smith K A, et al. Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning.[C] // Robotics: Science and Systems XIV : Vol 14. 2018.

[4] Todorov E, Erez T, Tassa Y. MuJoCo: A physics engine for model-based control[C] // 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. 2012 : 5026-5033.

[5] Savva M, Malik J, Parikh D, et al. Habitat: A Platform for Embodied AI Research[C] // 2019 IEEE/CVF International Conference on Computer Vision (ICCV). 2019 : 9339-9347.

[6] Clavera I, Nagabandi A, Fearing R S, et al. Learning to Adapt: Meta-Learning for Model-Based Control[J/OL]. CoRR, 2018, abs/1803.11347. http://arxiv.org/abs/1803.11347.

[7] Du Y, Narasimhan K. Task-Agnostic Dynamics Priors for Deep Reinforcement Learning[J/OL]. CoRR, 2019, abs/1905.04819. http://arxiv.org/abs/1905.04819.

[8] Xie A, Ebert F, Levine S, et al. Improvisation through Physical Understanding: Using Novel Objects As Tools with Visual Foresight[C] // Robotics: Science and Systems XV : Vol 15. 2019.

[9] Shi X, Chen Z, Wang H, et al. Convolutional LSTM Network: a machine learning approach for precipitation nowcasting[C] // NIPS'15 Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1 : Vol 28. 2015 : 802-810.

[10] Laversanne-Finot A, Péré A, Oudeyer P-Y. Curiosity Driven Exploration of Learned Disentangled Goal Spaces[C] // CoRL 2018 - Conference on Robot Learning. 2018 : 487-504.

[11] Pu Y, Gan Z, Henao R, et al. Variational Autoencoder for Deep Learning of Images, Labels and Captions.[C/OL] // Lee D D, Sugiyama M, von Luxburg U, et al. NIPS. 2016 : 2352-2360. http://dblp.uni-trier.de/db/conf/nips/nips2016.html#PuGHYLSC16.

[12] Forestier S, Mollard Y, Oudeyer P. Intrinsically Motivated Goal Exploration Processes with Automatic Curriculum Learning[J/OL]. CoRR, 2017, abs/1708.02190. http://arxiv.org/abs/1708.02190.

[13] Peng X B, Abbeel P, Levine S, et al. DeepMimic: example-guided deep reinforcement learning of physics-based character skills[J]. ACM Transactions on Graphics, 2018, 37(4) : 143.

[14] Peng X B, Kanazawa A, Malik J, et al. SFV: reinforcement learning of physical skills from videos[J]. ACM Transactions on Graphics, 2019, 37(6) : 178.

[15] Epshteyn A, Vogel A, DeJong G. Active reinforcement learning.[C/OL] // Cohen W W, McCallum A, Roweis S T. ACM International Conference Proceeding Series, Vol 307 : ICML. [S.l.] : ACM, 2008 : 296-303. http://dblp.uni-trier.de/db/conf/icml/icml2008.html#EpshteynVD08.

[16] Ho J, Ermon S. Generative Adversarial Imitation Learning.[C/OL] // Lee D D, Sugiyama M, von Luxburg U, et al. NIPS. 2016 : 4565-4573. http://dblp.uni-trier.de/db/conf/nips/nips2016.html#HoE16.

[17] Duan Y, Andrychowicz M, Stadie B C, et al. One-Shot Imitation Learning.[C/OL] // Guyon I, von Luxburg U, Bengio S, et al. NIPS. 2017 : 1087-1098. http://dblp.uni-trier.de/db/conf/nips/nips2017.html#DuanASHSSAZ17.

[18] Li S, Sun Y, Liu S, et al. Active physical inference via reinforcement learning.[J]. Cognitive Science, 2019 : 2126-2132.

[19] Baranes A, Oudeyer P-Y. Active learning of inverse models with intrinsically motivated goal exploration in robots[J]. Robotics and Autonomous Systems, 2013, 61(1) : 49-73.

[20] Brockman G, Cheung V, Pettersson L, et al. OpenAI Gym[J], 2016.

[21] Delhaisse B, Rozo L, Caldwell D G. PyRoboLearn: A Python Framework for Robot Learning Practitioners[C/OL] // Conference on Robot Learning (CoRL). 2019. https://github.com/robotlearn/pyrobolearn.

[22] Paszke A, Gross S, Massa F, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library[C] // Advances in Neural Information Processing Systems : Vol 32. 2019 : 8026-8037.

[23] Konda V R, Tsitsiklis J N. Actor-critic algorithms[M]. 2002.

[24] Fujimoto S, van Hoof H, Meger D. Addressing Function Approximation Error in Actor-Critic Methods[J/OL]. CoRR, 2018, abs/1802.09477. http://arxiv.org/abs/1802.09477.

[25] Sutton R, Barto A. Reinforcement Learning: An Introduction[M]. 1988.

[26] Andrychowicz M, Wolski F, Ray A, et al. Hindsight Experience Replay[J/OL]. CoRR, 2017, abs/1707.01495. http://arxiv.org/abs/1707.01495.

[27] Tang H, Houthooft R, Foote D, et al. #Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning[J/OL]. CoRR, 2016, abs/1611.04717. http://arxiv.org/abs/1611.04717.

[28] Gionis A, Indyk P, Motwani R. Similarity Search in High Dimensions via Hashing[C/OL] // VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1999 : 518-529. http://portal.acm.org/citation.cfm?id=671516.

[29] Charikar M. Similarity Estimation Techniques from Rounding Algorithms.[C/OL] // STOC. 2002 : 380-388. http://dblp.uni-trier.de/db/conf/stoc/stoc2002.html#Charikar02.

[30] Stadie B C, Levine S, Abbeel P. Incentivizing Exploration In Reinforcement Learning With Deep Predictive Models[J/OL]. CoRR, 2015, abs/1507.00814. http://arxiv.org/abs/1507.00814.

[31] Coumans E, Bai Y. Pybullet, a python module for physics simulation for games, robotics and machine learning[J], 2016.

[32] Wise M, Ferguson M, King D, et al. Fetch & Freight : Standard Platforms for Service Robot Applications[C] // . 2016.

[33] Eysenbach B, Salakhutdinov R, Levine S. Search on the Replay Buffer: Bridging Planning and Reinforcement Learning[J/OL]. CoRR, 2019, abs/1906.05253. http://arxiv.org/abs/1906.05253.

[34] Stadie B C, Yang G, Houthooft R, et al. The Importance of Sampling in Meta-Reinforcement Learning[C] // NIPS'18 : Proceedings of the 32nd International Conference on Neural Information Processing Systems. Red Hook, NY, USA : Curran Associates Inc., 2018 : 9300–9310.

[35] Kang Y, Kim D, Kim K. URDF Generator for Manipulator Robot.[C/OL] // IRC. [S.l.] : IEEE, 2019 : 483-487. http://dblp.uni-trier.de/db/conf/irc/irc2019.html#KangKK19.

[36] van Hasselt H, Guez A, Hessel M, et al. Learning functions across many orders of magnitudes[J/OL]. CoRR, 2016, abs/1602.07714. http://arxiv.org/abs/1602.07714.

[37] Xu B, Wang N, Chen T, et al. Empirical Evaluation of Rectified Activations in Convolutional Network[J/OL]. CoRR, 2015, abs/1505.00853. http://arxiv.org/abs/1505.00853.

[38] Nwankpa C, Ijomah W, Gachagan A, et al. Activation Functions: Comparison of trends in Practice and Research for Deep Learning[J/OL]. CoRR, 2018, abs/1811.03378. http://arxiv.org/abs/1811.03378.

[39] Lippmann R. An introduction to computing with neural nets[J/OL]. IEEE ASSP Magazine, 1987, 4(2) : 4-22. http://dx.doi.org/10.1109/MASSP.1987.1165576.

[40] Novikoff A B. On Convergence Proofs on Perceptrons[C] // Proceedings of the Symposium on the Mathematical Theory of Automata : Vol 12. New York, NY, USA : Polytechnic Institute of Brooklyn, 1962 : 615-622.

哈尔滨工业大学学位论文原创性声明和使用权限

学位论文原创性声明

本人郑重声明：此处所提交的学位论文《基于虚拟物理仿真思考的开放任务求解》，是本人在导师指导下，在哈尔滨工业大学攻读学位期间独立进行研究工作所取得的成果，且学位论文中除已标注引用文献的部分外不包含他人完成或已发表的研究成果。对本学位论文的研究工作做出重要贡献的个人和集体，均已在文中以明确方式注明。

作者签名：　　日期：　　年　　月　　日

学位论文使用权限

学位论文是研究生在哈尔滨工业大学攻读学位期间完成的成果，知识产权归属哈尔滨工业大学。学位论文的使用权限如下：

（1）学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文，并向国家图书馆报送学位论文；（2）学校可以将学位论文部分或全部内容编入有关数据库进行检索和提供相应阅览服务；（3）研究生毕业后发表与此学位论文研究成果相关的学术论文和其他成果时，应征得导师同意，且第一署名单位为哈尔滨工业大学。

保密论文在保密期内遵守有关保密规定，解密后适用于此使用权限规定。

本人知悉学位论文的使用权限，并将遵守有关规定。

作者签名：　　日期：　　年　　月　　日

导师签名：　　日期：　　年　　月　　日

致　谢

衷心感谢导师 范晓鹏 教授对本人的精心指导。他的言传身教将使我终生受益。

感谢哈工大 LaTeX 论文模板 hiThesis 。